Methodologies for Model-Driven Development of Adaptive Web Applications: An Analytical Survey

Mona Fadavi*, Raman Ramsin
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran.

* Corresponding author. Email: mfadavi@ce.sharif.edu
Manuscript submitted August 15, 2015; accepted October 23, 2015.
doi: 10.17706/jsw.11.1.94-109

Abstract: Due to the rapid expansion of web applications, the information and services provided on the web have proliferated, leading to ever-increasing complexity. This has not only resulted in the utmost significance of information quality and accuracy on the web, but has also necessitated that access to information be improved. Hence, adaptive web systems have emerged, which focus on adapting web content, presentation, and navigation to meet the needs of the users according to their individual circumstances and preferences. In this field, focus has gradually shifted from creation of new adaptive techniques to solving the problems of analysis and design of adaptive applications; in other words, it has become important to handle the dynamism of the runtime environment and the complexity involved in developing these applications. Model-Driven Engineering (MDE) is considered a promising approach for overcoming these problems, mainly due to its modeling features: Models are created at different levels of abstraction (thus enhancing complexity management), and the process can potentially be automated through the use of transformation rules. We provide a review of several prominent methodologies that utilize MDE for developing adaptive web applications, and propose a criteria-based evaluation approach that highlights their strengths and weaknesses. Evaluation results can be used for comparing and selecting methodologies for use in web development projects; they can also be used for constructing a new methodology that exploits the strengths and addresses the shortcomings of existing methodologies.

Key words: Adaptive web application, criteria-based evaluation, model-driven engineering, software development methodology.

1. Introduction

In companies that advance their businesses by providing web applications and web services, achieving high quality is of utmost importance, since a desirable return-on-investment is only possible through increasing the number of returning customers. Due to the diversity of the potential users of web applications, developers have to cope with the challenges of unknown expectations and behavioural patterns of the users. On the other hand, the large volume of information on the web has led to difficulties in finding the desired information in hyperspace. Therefore, in order to keep the customers satisfied, web applications have turned to observing the users' actions and then providing them with the appropriate information by filtering out the irrelevant information and taking into account the users' preferences. This has led to the advent of adaptive web applications, which can adapt web content, navigation, and presentation according to the users' circumstances and preferences. Adaptive web applications are a subset of adaptive hypermedia systems, which "build a model of the goals, preferences and knowledge for each individual user and apply it throughout the interaction for adaptation to the needs of the users" [1].

Model-Driven Engineering (MDE) is considered as a powerful approach for developing complex software systems. It promotes the production of models through model transformation, ultimately leading to the generation of executable code. The main process in MDE includes the definition of a set of models, along with a set of rules for (semi)automatic transformation of the models into one another. MDE can improve reusability and portability, and lower development costs.

Due to the growing need for adaptive web applications, an engineering methodology is required for developing these systems to ensure high quality and also to assist the developers in building these applications systematically. Generating adaptive responses to user requests is a typically complex process. In order to manage this complexity, it is necessary to augment the application models with adaptivity features. Consequently, a description of runtime behaviour is required in order to identify how to use the information and relationships to achieve adaptation. MDE seems to be an appropriate approach for overcoming the complexity and runtime dynamism involved in developing these applications, as it supports complexity management by prescribing separate models at different levels of abstraction, and enables the developers to (re)produce the models (semi)automatically through the use of transformation rules.

In this paper, we review six prominent model-driven adaptive web development methodologies; these are the only methodologies that fully cover the three areas targeted (adaptivity, web development, and MDE) while providing the documentation required for scrutinizing the methodologies. We also propose a special set of evaluation criteria, which when applied to these methodologies, highlight their strengths and weaknesses. The evaluation criteria and results can be used by developers to assess, compare and select their required model-driven adaptive web development methodology. They can also be used by method engineers to create a new methodology by exploiting the strong features and resolving the shortcomings of existing methodologies.

The rest of this paper is structured as follows: A brief review of six model-driven adaptive web development methodologies is presented in Section II; Section III introduces the proposed set of evaluation criteria; Section IV analyzes the results of applying the criteria to the methodologies; and Section V presents the conclusions and suggests ways for furthering this research.

2. Model-Driven Adaptive Web Development Methodologies

In this section, we review the high-level process of the six methodologies based on the process-centered template introduced in [2], particularly focusing on the support provided for adaptivity in each methodology.

2.1. Extended OOWS Methodology

Rojas [3] has proposed an extension to the Object-Oriented Web Solution (OOWS) methodology in order to make it applicable to adaptive web development. The model-driven process of OOWS has been extended by enhancing requirements specification and conceptual modeling to support the modeling of adaptive web applications (as shown in Fig. 1). It prescribes exhaustive user modeling which captures the various aspects of users in user diagrams and navigational behaviour diagrams. The main contribution of this methodology is applying implementation-level adaptive techniques to conceptual-level modeling. This is done by providing high-level adaptive techniques in terms of conceptual primitives (called adaptive primitives) in the navigational schema.

Transformation is performed from each possible occurrence of adaptive requirements to the corresponding navigational descriptions, so that for each adaptive requirement, a set of different modeling strategies is introduced in terms of adaptive primitives. This idea rises from the observation that every adaptation method can be implemented with different techniques, so the implementation variant of the adaptive primitives is determined by fulfilment of constraints for a given user (based on the user model). In the OOWS methodology, the development process consists of two major stages: Problem Specification and Solution Development. In Extended OOWS, only the Problem Specification stage has been enhanced in order to support adaptivity. This has been done by extending the following two steps:

2.1.1. Requirements specification

In this step, two complementary activities are already prescribed in the OOWS methodology: (1) functional requirements are specified in a mission statement, a functional refinement tree, and a use-case diagram; (2) navigational requirements are specified in a task diagram, task specifications, and data description templates [4]. In Extended OOWS, user stereotype diagrams and user specification templates have been added in order to model user-related requirements. Also, the task diagram, task specifications, and data description templates have been enhanced for specifying adaptive requirements. Task specifications have been enhanced by determining adaptive tasks and augmenting the relevant task specifications so that for each adaptive task, an activity diagram is defined which depicts the constraints imposed on accessing the nodes, operations, and links that are included in the adaptive navigational requirements. The data description has also been extended by introducing constraints on accessing the attributes of each node.

2.1.2. Conceptual modeling

In this step, the structural schema, dynamic schema, functional schema, navigational schema, and presentation schema are produced as prescribed in the OOWS methodology [4]. In Extended OOWS, a user schema is also produced as an augment to the structural schema, thus adding descriptions of users and their behaviour to the structural model. Also, the existing OOWS navigational schema has been extended with adaptive primitives that allow the specification of adaptive navigation and presentation. Based on these primitives, a set of modeling strategies are defined that guide the designers in incorporating adaptive methods through the use of adaptive techniques.

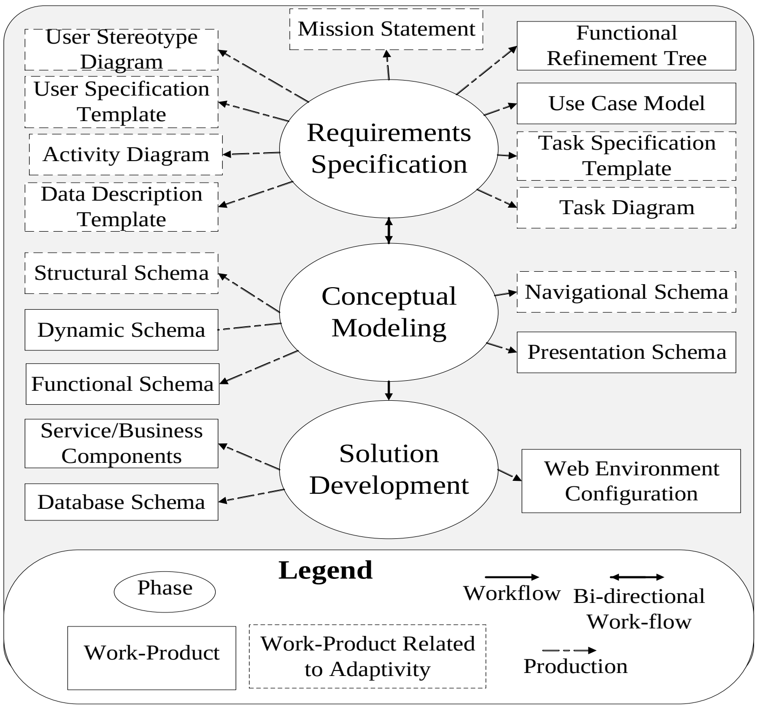

Fig. 1. Extended OOWS process.

2.2. Extended WebML Methodology

Ceri *et al*. [5] have proposed a methodology for developing context-aware multichannel web applications by extending the Web Modeling Language (WebML) methodology with adaptive actions that are triggered by the context. This extension addresses the main concerns in developing such systems by answering three questions: (1) How should we describe and manage context data during data design? (2) How should we combine non-adaptive hypermedia with adaptive hypermedia during hypertext design? (3) How should we attach adaptive actions to the hyperspace in reaction to context changes? The tasks prescribed by Extended WebML for supporting adaptivity (shown in Fig. 2) are explained below:

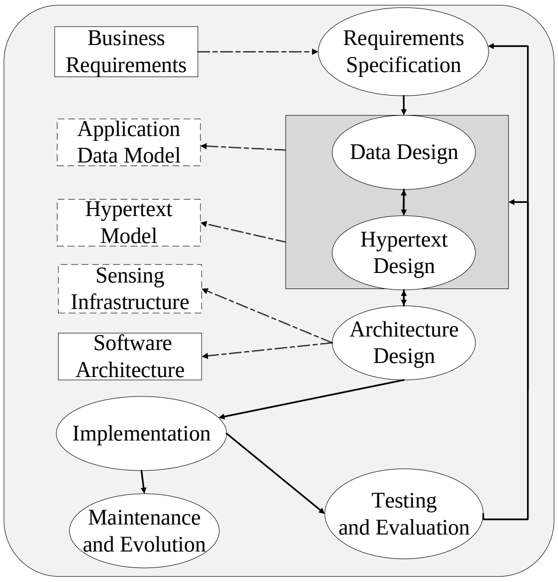

Fig. 2. Extended WebML process.

2.2.1. Requirements specification

WebML has prescribed the elicitation of personalization requirements, but Extended WebML has failed to provide a specific method for extracting the set of adaptive requirements (along with their context triggering properties) that is required in context-aware web applications.

2.2.2. Data design

Extended WebML prescribes to enrich the application's data sources with context properties. To this aim, it prescribes the addition of user profile data, dynamic user model data, and context data, thus extending the data schema with three sub-schemas.

2.2.3. Hypertext design

Extended WebML prescribes to: 1) Tag adaptive pages with a 'C' label, 2) attach context-clouds to adaptive pages, which represent the adaptive logic and the set of adaptivity actions associated with the pages, 3) indicate a polling interval for each adaptive page which shows the querying time on the context model, 4) identify an adaptation policy for each adaptive page to determine the priority of the users or the context for triggering adaptive actions, and 5) use new modeling concepts and primitives for adaptive actions, which allow visual specification of actions for context model management and hypertext adaptation. In addition, WebML primitives are also used for adaptive actions.

2.2.4. Architecture design

Certain constraints, related to context-model management, must be captured in the architecture. The relevant tasks are as follows:

a) Context data acquisition: A client context-sensing infrastructure should be designed to sense the parameters on the client-side and send them back to the application. Also, a centralized sensing infrastructure is needed for handling server-side parameters.

b) Context model updating: It is necessary to update the context model at the data level.

c) Context model monitoring: Two methods are proposed: In [5], monitoring is achieved by periodically refreshing the pages at any user's request; whereas in [6], a monitoring module is introduced which refreshes the pages when context variation triggers the adaptive action.

In addition, Extended WebML prescribes a run-time algorithm that supports automatic execution of adaptive actions; the page computation algorithm of WebML is still used for non-adaptive actions.

2.2.5. Implementation

Two prototypes are prescribed: The first prototype provides a solution for implementing the context-aware features that can be obtained without altering WebML's run-time environment; the second prototype is produced by the WebRatio CASE tool to fully reflect the proposed visual design method. These prototypes are complementary, and it is possible to design and generate their code from extended design models.

2.2.6. Testing and evaluation, and maintenance and evolution

Performed as their Webml counterparts. In addition to the above, a method for modeling user navigational behaviour has been proposed in [7], which allows performing adaptive actions in response to predefined navigation patterns. It uses a Web Behaviour Model (WBM), consisting of a finite state automaton and high-level event-condition-action rules. These rules allow the interpretation of events as user requests, conditions as WBM scripts, and actions as a set of adaptive actions expressed in terms of a chain of WebML operation units.

2.3. UM-MAIS Methodology

UM-MAIS [8] incorporates a full-lifecycle process for developing multi-channel adaptive web information systems. It integrates and specializes three prominent methods for requirements engineering, hypertext design, and service design. Its main aim is to exploit the strengths of these three methods in a way that supports both model-driven and agile development. The major advantage of this methodology is combining hypertext, data, and service designs so that their associations and interactions can be utilized when design problems arise; e.g., when some activities cannot be fully addressed by hypertext design, the developer can resort to service design (via external operation invocation), or to data design (via inclusion of new data in the data schema). The phases of UM-MAIS (shown in Fig. 3) are explained below:

2.3.1. Requirements management

UM-MAIS extends the AWARE method [9] for providing goal-oriented requirements modeling to support adaptivity and web service integration. The following steps are performed:

a) Requirements elicitation: In accordance with the refinement process of AWARE, high-level goals are refined into sub-goals, and eventually into requirements.

b) Requirements analysis and classification: The requirements are first classified into data, service, and hypertext requirements. Certain predefined questions are then answered in order to determine the impact of context factors on the requirements. Based on these answers, the associations among requirements and context dimensions are discovered; the resulting adaptivity requirements are added to the requirements set. Moreover, the relationships among the goals, requirements and produced design artefacts are determined in order to support traceability.

2.3.2. High-level design

This phase contains two nested sub-phases, which are explained below:

a) Data Design: Performed according to the data design activity prescribed in extended-WebML [5].

b) High-Level Hypertext and Service Design: High-level hypertext design is performed according to steps 1 to 4 of the Extended-WebML's hypertext design activity [5]. In addition, the hypertext designer must determine the pages that are needed for service invocations, and should then send the list of uncovered requirements to the service designer, who requires this list along with a data schema and the shared knowledge of the application domain as input. Service design is then performed according to the WSMoD methodology [10], in which the services are designed based on specific quality-of-service features. Ultimately, the results are sent to the hypertext designer to be used in low-level hypertext design.

2.3.3. Low-level hypertext and service design

Low-level hypertext design is performed according to step 5 of the hypertext design activity of Extended-WebML [5]. Low-level service design involves the following steps:

a) Adaptability design: Two activities are performed, through which system models are refined and enriched with adaptivity-related information: 1) in the data and operation design activity, the class diagram of the services is enriched, and 2) in the interaction design activity, the interaction diagrams related to user, channel, and quality-of-service requirements are enriched.

b) Customization: Through the two activities of channel and user customization, service specifications are validated with respect to actual channel and user profiles.

c) Web service description: The diagrams obtained so far are transformed into web service descriptions using standard languages such as WSDL and WSOL.

2.3.4. Development and deployment

UM-MAIS prescribes certain tools for each phase; however, the main development starts by WebML's run-time environment which generates XML documents from a WebML schema. UM-MAIS's rule-engine takes these documents as input and generates HTML pages adapted to the current context. Deployment is supported by WebML's run-time tool and other web service development tools.

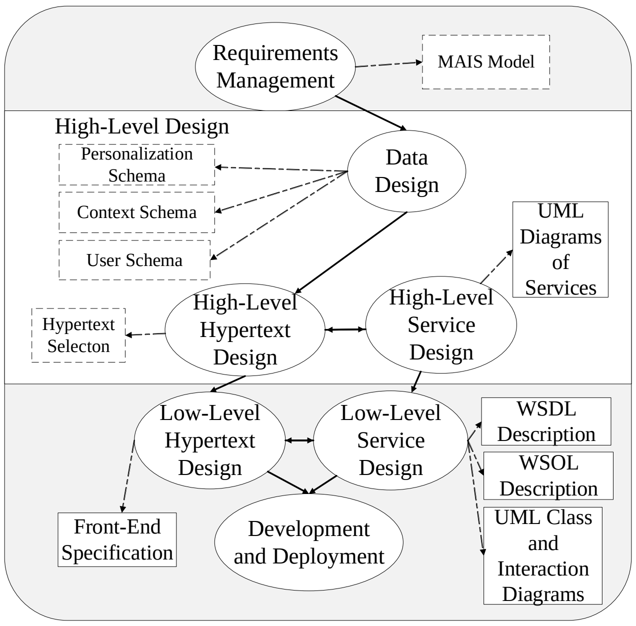

Fig. 3. UM-MAIS process.

2.4. Hera Methodology

Hera [11] aims to address the development of Web Information Systems (WIS). The main feature of WIS is its focus on gathering information from heterogeneous sources, and semi-automatic generation of the web presentation in response to user queries. Hera's process uses the Resource Description Format (RDF) as the main format for the data used in the transformation process, and adopts XSLT as its transformation approach. The phases of Hera (shown in Fig. 4) are explained below:

2.4.1. Conceptual design

Domain data is described in a conceptual model based on RDF. It consists of concepts, attributes and relationships, but no operations.

2.4.2. Integration design

This phase has two main activities, which are explained below:

a) Integration: Responsible for mapping data from different heterogeneous sources to the concepts of source ontologies, then relating them to the corresponding concepts in the conceptual model of the target WIS.

b) Data retrieval: Elicits user-formulated queries and translates them into queries on the conceptual model. In response to the resulting query, the desired data are provided and converted into conceptual model instances.

c) Application Design: In this phase, navigation views are described in the application model, which consists of ovals (domain concepts), slices (navigational views on concepts), links, attributes, and operations.

2.4.3. Adaptation design

Hera realizes adaptation by attaching inclusion conditions to conceptual, application, and presentation model elements, but it interprets adaptation as either *adaptability*, for when the conditions are based on user/platform profiles, or *adaptivity*, for when they are based on the user model. These conditions govern the visibility of the attributes or relationships of the model elements, leading to conditional inclusion of fragments and execution of link-hiding techniques.

2.4.4. Presentation design

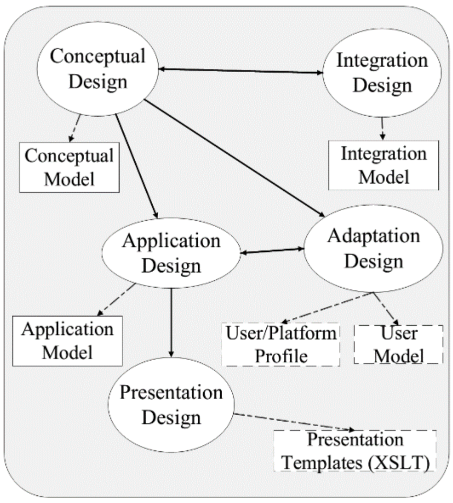

Fig. 4. Hera process.

Based on the data passed from the previous phase, the presentation is generated through the following transformation steps:

a) Transformation specification generation: Transformation specifications are generated based on adaptation and application model elements.

b) Application model instance generation: Based on the resulting transformation specifications, the conceptual model is converted into an application model.

c) Presentation generation: Presentation is generated as specific to the format of the user platform.

2.5. A-OOH Methodology

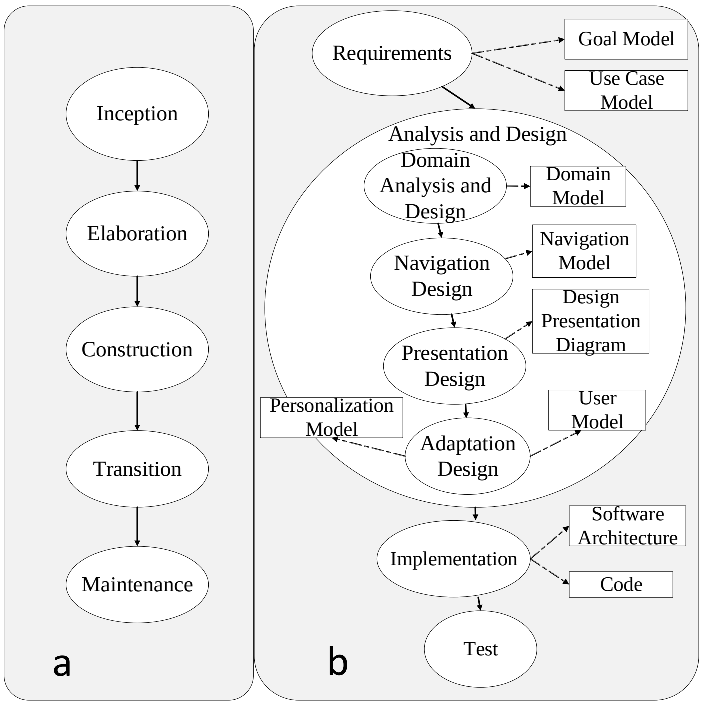

Fig. 5. A-OOH process: a) macro process, b) micro process.

A-OOH [12] is an extension to the OO-H methodology for supporting adaptivity concerns. A-OOH is based on the Unified Process (UP) methodology, and its lifecycle (macro process) consists of five sequential phases: Inception, Elaboration, Construction, Transition, and Maintenance. The process incorporates four workflows, executed iteratively throughout the phases (constituting the micro process): requirements, analysis and design, implementation, and test (shown in Fig. 5). Adaptation is applied based on a set of Event-Condition-Action (ECA) rules expressed in a high-level language, called the Personalization Rule Modeling Language (PRML), which specifies personalization at the design level. Although PRML has been used in the A-OOH methodology, it is independent of any particular web modeling methodology or environment; therefore, personalization specifications expressed in PRML are portable across different development environments. In order to achieve this level of portability, certain prerequisites have been defined for any web design methodology to which personalization by PRML is to be added, and a special method has been proposed in order to specify the strategies of PRML-assisted personalization in different methodologies. Furthermore, due to the conformance of PRML with the MOF metamodel, its portability has been demonstrated by defining a set of transformation rules from PRML to two well-known methodologies: UWE and Hera. Thus, the designer can define personalization via PRML, and then either perform transformation to produce the artifacts of the desired methodology, or use the specifications directly (which requires a rule engine that supports PRML). The main advantages of A-OOH are its ability to define the personalization strategies as reusable specifications, and its prototyping tool (AWAC), which fully supports the automatic generation and management of adaptive web applications based on the A-OOH process. The workflows of A-OOH are explained below:

2.5.1. Requirements

Similar to OO-H, a use-case diagram is used for capturing adaptation requirements, although using the i* framework has also been proposed (in [13]); thus, navigation, service, personalization, and layout requirements are specified in terms of task stereotypes, and data requirements are defined in terms of resource stereotypes (*resource* and *task* are i* concepts).

2.5.2. Analysis and design

This workflow consists of four nested stages, as explained below:

a) Domain analysis and design: In accordance with OO-H, domain modeling produces a UML class diagram.

b) Navigation design: A navigation model is produced, which represents navigation nodes in terms of: 1) navigational classes (views on domain classes), 2) navigational targets (grouped model elements that collaborate for satisfying the navigation requirements of the user), and 3) navigational collections (hierarchical structures of navigation classes or targets). Also, two types of navigation links are defined: 1) service link, representing the activation of operations, and 2) traversal link, representing the paths among navigation nodes.

c) Presentation design: A design presentation diagram is produced which enriches the navigation model by adding two abstraction levels. At the first level, concepts related to the abstract structure of the website are defined by grouping navigation nodes into abstract pages; designers can add new static pages at this level. At the second level, specific presentation details, as to layout and style, are determined for the abstract pages.

d) Adaptation Design: This stage includes two types of modeling, user modeling and personalization modeling, the details of which are explained below:

User modeling: The user model can be defined for each individual user or user group, and its data is classified into four categories: 1) user characteristics, such as disabilities, age, and hobbies, 2) user requirements, which describe the specific goals of the user's interaction with the system, 3) user context, which consists of the information concerning the context of the current session, and 4) user browsing behaviour. Also, domain-dependent information is elicited and documented.

Personalization modeling: The set of ECA rules that constitute the personalization model are expressed in PRML. The rules are classified into three types: 1) acquiring personalization data, 2) classifying users based on their profiles, and 3) performing the desired adaptation.

2.5.3. Implementation

The software architecture is determined, and external data and web services are specified. This ultimately leads to automatic generation of the web interface by using the AWAC tool.

2.5.4. Test

Evaluation activities are performed, including prototyping, verification, and validation.

2.6. Extended UWE Methodology

UWE [14] focuses on systematic development of web applications based on OMG standards, model-driven principles, and computer-aided tools. This methodology has evolved over time, so that it now supports contemporary features of web systems such as security, rich internet applications, and adaptivity. UWE's MDE approach proposes meta-models for all the models produced. It also defines a full set of transformation rules, and provides a semi-automatic process for tool-supported model design, model transformation, model checking, and automatic generation of web applications. Extended UWE [15], [16] extends UWE with adaptive features, incorporated as orthogonal concerns into the different web-related levels (navigation, content, and presentation), methodology phases, and modeling aspects (structural, functional, and behavioural). Extended UWE follows an aspect-oriented approach, defining model and run-time aspects to support static/dynamic aspect weaving.

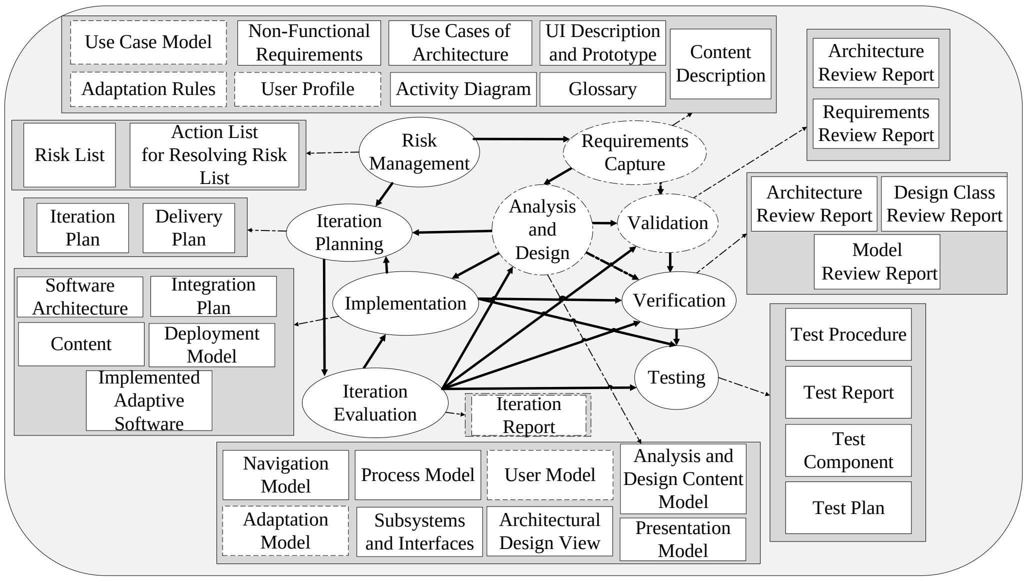

Fig. 6. Extended UWE: micro process.

The process of Extended UWE is based on the UP methodology. The macro process consists of five sequential phases: Inception, Elaboration, Construction, Transition, and Maintenance. In each phase, three sets of workflows are executed iteratively, thus forming the micro process: development, project management, and quality management (shown in Fig. 6). The *project management* set consists of three workflows: risk management, iteration planning, and iteration evaluation. The *quality management* set includes three workflows: validation, verification, and testing. The three workflows of the main set, *development,* are explained below:

2.6.1. Requirements capture

Specific techniques are prescribed for requirements elicitation and validation. Moreover, light-weight versions of UML use-case and activity diagrams are used for requirements specification.

2.6.2. Analysis and design

Various modeling activities are performed. UML class diagrams are used for structural domain modeling. User and environment properties are captured in a user model. Navigation is modeled in light-weight class diagrams, in which the hypertext structure (nodes and links) is depicted. Access primitives are defined for complex constructs. Process flows are captured in activity diagrams; as they must also be integrated in the navigation model, process classes are defined and associated with navigation classes. Presentation is modeled based on the navigation model to depict the UI elements of the web pages. Light-weight class and interaction diagrams are used for modeling the structure and behavior of the presentation. Adaptation is modeled in terms of aspects, which are weaved into the models produced. Architectural design is also performed, and subsystems and their interfaces are defined.

2.6.3. Implementation

This workflow consists of implementation activities, content provision for the web application, and subsystems integration.

3. Qualitative Criteria for Evaluation of Model-Driven Adaptive Web Development Methodologies

We have evaluated the methodologies reviewed in the previous section through applying the feature analysis approach [17], which is widely used for evaluating software development methodologies. As this approach is based on qualitative evaluation, the development of a set of Qualitative Criteria (QC) has been an important objective of our research. It must be demonstrated that the proposed criterion set is detailed and comprehensive enough to reveal the strengths and weakness of methodologies. Therefore, the extracted criteria were evaluated by applying the set of meta-criteria proposed in [18]): comprehensiveness, accuracy, simplicity, consistency, minimal overlap, generality, and balance. To ensure the satisfaction of these meta-criteria, we have applied an iterative elicitation-evaluation process for identifying the QC [19]. We can therefore claim that our elicited QC are comprehensive enough to cover the important features of the target methodologies, accurate enough to discern the shortcomings and strengths of the methodologies, simple to understand and measure, consistent (as all inconsistencies have been resolved), minimally overlapped (as each criterion pertains to one independent category), general enough for application to all relevant methodologies, and properly balanced (as all domain concepts have been covered in their own right).

Table 1. Criteria for Evaluating Model-Driven Aspects

	Criterion Name	Type	Description of Levels
Tool-Related QC	CIM to PIM Transformation PIM to PSM Transformation	Scale	A: Not provided; B: Provided with tool support; C: Rules are defined but tool support is not provided.
	Automatic Code Generation Metadata Management Automatic Test Traceability between Models	Simple	I (Involved): The methodology explicitly provides the relevant support; D (Devolved): The methodology does not provide any support.
Model-Driven Architecture-Related QC	Tool Selection/Implementation	Scale	A: No specific toolset or explicit guidelines provided; B: Incomplete toolset or general guidelines provided; C: Complete toolset or precise guidelines provided.
	CIM Creation PIM Creation PSM Creation	Scale	A: Model production is not supported; B: General guidelines provided; C: Precise guidelines provided.
	Verification/Validation (V&V) Extension of Rules Round-Trip Engineering Source and Target Models Synchronization	Scale	A: The activity is not defined and is devolved to developers; B: The activity is not defined in detail; C: Explicit and detailed guidelines and techniques are provided for this activity.

Due to the fact that our target domain is the intersection of three fields (i.e., MDE, Adaptive Systems, and Web Engineering), our proposed evaluation framework consists of three sets of QC. The proposed QC are shown in Tables 1, 2, and 3, corresponding to model-driven aspects, web-engineering aspects, and adaptivity aspects, respectively. The first set, which is related to model-driven aspects, consists of the tool-related and architecture-related QC proposed in [20]. The second set, which covers web engineering aspects, has been obtained by refining the general QC proposed for evaluating software development methodologies proposed in [2] based on the web-critical requirements discussed in [21] and [22]; these QC are divided into four sets: general, development-related, management-related, and artifact-related. In order to define the third set, which pertains to adaptivity aspects, we have examined and adapted the sets of QC proposed in [23], [24], and [25].

In order to enhance the measurability of the QC, they have been defined in two possible forms: (1) Scale form (multilevel), for which the evaluation results are selected from among predefined discrete levels, each representing the degree of fulfilment of the criterion, and (2) Simple form, for which the results are of the "Yes/No" type, denoting the fulfilment or non-fulfilment of the criterion.

Table 2. Criteria for Evaluating Web Engineering Aspects

Criterion Name		Type	Description of Levels
Technological Properties Consideration		Simple	I (Involved): The methodology explicitly provides the relevant support; D (Devolved): The methodology does not provide any support.
Support of Short Development Cycles			
Coverage of User Requirements Variation			
Concurrency Support			
Organization of Multidisciplinary Teams			
Basis in Architecture			
Existence of Necessary Models for Web Development		Scale	A: Fully supported; B: Some models are supported; C: None supported.
Support of Exclusive Modeling Language		Scale	A: Exclusive modeling language is provided; B: Extended version of an existing language is used; C: No modeling language is prescribed.
Support of Exclusive Content Provision Activities		Scale	A: Exclusive web activities prescribed; B: General activities prescribed; C: No activities prescribed.
Support of Exclusive Test Activities			
Coverage of General Life Cycle	Requirements Engineering	Scale	A: Not supported; B: Supported with general guidelines; C: Supported with detailed directives.
	Analysis		
	Design		
	Implementation		
	Test		
	Deployment		
	Maintenance		
Coverage of Umbrella Activities	Project Management	Scale	A: Not supported; B: Supported with general guidelines; C: Supported with detailed directives.
	Risk Management		
	Quality Management		
Clarity of Development Process Definition		Scale	A: Work-products, actors and activities are completely supported and precisely described; B: Work-products, actors and activities are incompletely supported or just mentioned; C: Work-products, actors, and activities are weakly supported.
Seamlessness and Smoothness of Transition between Phases		Scale	A: Only seamlessness provided; B: Only smoothness provided; C: Both provided; D: None provided.
Basis in Functional and Non-Functional Requirements		Scale	A: Both supported; B: Non-functional requirements supported; C: No support.
Product Dependencies Specification		Scale	A: Product dependencies are not specified; B: Few products are evolved or reused in other phases; C: Most products are evolved or reused in other phases; D: Products are not interdependent.
Traceability to Requirements		Simple	I (Involved): The methodology explicitly provides the relevant support; D (Devolved): The methodology does not provide any support.
Encouragement of Active User Involvement			
Complexity Management			
Practicability & Practicality			
Scalability			
Flexibility			
Configurability & Extensibility			
Support of Formal Modeling Facilities			
Tangibility of Artifacts		Scale	A: Artifacts are tangible for both users and developers; B: Artifacts are tangible for users only; C: Artifacts are tangible for developers only; D: Artifacts are not tangible.
Covering Various Aspects of Modeling		Scale	A: All aspects supported; B: Most aspects strongly supported; C: Most aspects weakly supported.

4. Analysis of the Evaluation Results

As the model-driven evaluation results show (Table 4), only the A-OOH, Extended OOWS, Extended UWE, and Hera methodologies support transformation from CIM to PIM. As for tool support, Extended WebML and UM-MAIS use an extension of the WebML run-time environment, UM-MAIS proposes detailed guidelines for selection of tools for each of its phases, and A-OOH and Hera provide exclusive tools (the AWAC prototype, and the XSLT dedicated processor and engine). Most of the methodologies are weak in supporting round-trip engineering, extension of rules, V&V, metadata management, traceability, and automatic testing.

Table 3. Criteria for Evaluating Adaptivity Aspects

Criterion Name	Type	Description of Levels
Decoupling Context from Business Logic General Scope of Adaptive Systems Support Reusability of Adaptation Rules	Simple	I (Involved): The methodology explicitly provides the relevant support; D (Devolved): The methodology does not provide any support.
Context Modeling	Scale	A: All the main properties of the context (user, environment and browsing history) are modeled; B: Some properties of the context are modeled; C: Not supported.
Adaptivity-Level Support	Scale	A: Fully supported; B: Some aspects supported; C: Not supported.
Adaptation Methods and Techniques Support	Scale	A: Mainly supported; B: Partially supported; C: Not supported.
Adaptation Considerations in Development Process	Scale	A: Work-products, actors and activities are completely supported and precisely described; B: Work-products, actors and activities are incompletely supported or just mentioned; C: Work-products, actors, and activities are weakly supported.
Support of Various Adaptation Triggers	Scale	A: Only interests and preferences of users are considered for triggering adaptive actions; B: In addition to A, environmental properties are also considered; C: In addition to B, navigation styles, user groups and other factors are also considered.

As the web-engineering evaluation results show (Table 5), Extended WebML, Extended UWE and A-OOH support full-lifecycle development. Nevertheless, only Extended UWE supports umbrella activities. A-OOH, Extended WebML and Extended UWE incorporate iterative-incremental processes, so they can successfully cope with requirements mutability. However, none of the methodologies supports short development cycles. As for the modeling language used, A-OOH uses PRML as its exclusive language, but others use existing languages.

As the adaptivity evaluation results show (Table 6), Hera is the only methodology that does not support the general scope of adaptive systems, as it is only intended for WIS development. Extended OOWS is the only methodology that strongly supports context modeling; it is also the only methodology that strongly supports adaptation methods and techniques, so much so that adaptive primitives are defined for each method and technique. In Extended WebML, methods are just mentioned for adaptive requirements extraction, but in Extended OOWS, A-OOH, Extended UWE, and UM-MAIS, adaptivity-related activities are precisely defined; in contrast, Hera does not properly cover these activities, and just prescribes a few modeling activities. Since A-OOH uses PRML for specifying adaptive rules, it is the only methodology in which the rules can be reused.

Overall, it can be observed that:

1) Most of these methodologies produce the CIM and PIM. PSM, however, is mostly neglected.
2) Most of these methodologies do not explicitly define a set of transformation rules based on metamodels; their transformation approaches are therefore vague and excessively tool-dependent.
3) Most of these methodologies do not support the whole spectrum of adaptation methods and techniques; adaptivity suffers as a result.
4) Critical requirements such as support for short development cycles and requirements mutability are

not addressed in most methodologies.

Table 4. Results of Applying Model-Driven QC

Methodology \ Criterion	CIM Creation	PIM Creation	PSM Creation	V&V	Tool Selection/ Implementation	Extension of Rules	Source and Target Models Synchronization	Round-trip Engineering	CIM to PIM Transformation	PIM to PSM Transformation	Automatic Code Generation	Automatic Test	Metadata Management	Traceability between Models
Extended OOWS	C	C	A	A	A	A	A	A	C	A	D	D	D	D
Extended WebML	B	C	A	A	C	A	A	A	A	A	I	I	D	D
UM-MAIS	C	C	A	A	C	A	A	A	A	A	I	I	D	D
Hera	A	C	C	A	C	A	A	A	B	B	I	D	D	D
A-OOH	C	C	A	A	B	A	A	A	C	A	I	D	D	D
Extended UWE	C	C	C	C	C	A	C	A	B	B	I	I	D	D

Table 5. Results of Applying Web Engineering QC

Methodology \ Criterion	Coverage of General Life Cycle							Coverage of Umbrella Activities			Clarity of Development Process Definition	Seamlessness and Smoothness of Transition between Phases	Basis in Functional and Non-Functional Requirements	Product Dependencies Specification	Traceability to Requirements	Encouragement of Active User Involvement	Complexity Management	Practicability & Practicality	Flexibility	Scalability	Extensibility & Configurability	Support of Formal Modeling Facilities	Tangibility of Artifacts	Covering Various Aspects of Modeling	Support of Exclusive Content Provision Activities	Support of Exclusive Test Activities	Support of Short Development Cycles	Coverage of User Requirements Variation	Basis in Architecture	Concurrency Support	Organization of Multidisciplinary Teams	Existence of Necessary Models for Web Development	Support of Exclusive Modeling Language	Technological Properties Consideration
	Req. Engineering	Analysis	Design	Implementation	Test	Deployment	Maintenance	Project Management	Risk Management	Quality Management																								
Extended OOWS	C	C	C	C	A	A	A	A	A	A	B	B	A	B	I	D	D	I	D	D	D	D	A	A	C	C	D	D	D	I	D	A	B	D
Extended WebML	B	C	C	C	B	B	B	A	A	A	B	B	B	A	I	D	I	I	D	I	D	D	C	B	C	B	D	I	I	D	D	A	B	D
UM-MAIS	C	C	C	C	A	B	A	A	A	A	B	B	A	B	I	D	I	I	D	I	D	D	C	A	C	C	D	D	D	I	D	A	B	D
Hera	A	A	C	C	A	A	A	A	A	A	C	D	C	B	D	D	I	I	D	I	D	D	C	C	C	C	D	D	D	I	D	A	B	D
A-OOH	C	C	C	C	B	B	B	A	A	A	B	B	A	B	I	D	I	I	D	I	D	D	C	B	C	B	D	I	I	D	D	A	A	D
Extended UWE	C	C	C	C	C	B	C	B	B	B	A	B	A	C	I	D	I	I	D	I	D	I	A	A	B	B	D	I	I	D	I	A	B	D

Table 6. Results of Applying Adaptivity QC

Methodology \ Criterion	Decoupling Context from Business Logic	Reusability of Adaptation Rules	General Scope of Adaptive Systems Support	Context Modeling	Adaptation-Level Support	Adaptation Methods and Techniques Support	Support of Various Adaptation Triggers	Adaptation Considerations in Development Process
Extended OOWS	I	D	I	A	A	A	A	B
Extended WebML	I	D	I	B	A	B	C	B
UM-MAIS	I	D	I	B	A	B	A	A
Hera	D	D	D	B	B	C	B	C
A-OOH	I	I	I	B	A	B	A	B
Extended UWE	I	D	I	B	A	B	A	A

5. Conclusion and Future Work

We have reviewed six prominent model-driven adaptive web methodologies and have evaluated them through criteria-based assessment, thus highlighting their strengths and weaknesses. We aim to further this research by proposing a methodology which clearly defines comprehensive modeling levels (based on metamodels) and the corresponding transformation rules. It should also properly resolve the deficiencies of existing methodologies, as identified in this research.

References

[1] Brusilovsky, P. (1996). Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction, 6(2-3)*, 87-129.

[2] Ramsin, R., & Paige, R. F. (2008). Process-centered review of object-oriented software development methodologies. *ACM Computing Surveys, 40(1),* 1-89.

[3] Rojas, D. G. E. (2006). Modeling adaptive web applications in OOWS. PhD dissertation, University of Valencia, Spain.

[4] Pastor, O., Fones, J., & Pelechano, V. (2003). OOWS: A method to develop web applications from web-oriented conceptual models. *Proceedings of the International Workshop on Web Oriented Software Technology* (pp. 65-70).

[5] Ceri, S., Daniel, F., Matera, M., & Facca, F. M. (2007). Model-driven development of context-aware web applications. *ACM Transactions on Internet Technology, 7(1),* 1-33.

[6] Ceri, S., Daniel, F., Facca, F. M., & Matera, M. (2007). Model-driven engineering of active context-awareness. *World Wide Web, 10(4),* 387-413.

[7] Ceri, S., Daniel, F., Demaldé, V., & Facca, F. M. (2005). An approach to user-behavior-aware web applications. *Proceedings of the International Conference on Web Engineering* (pp. 417-428).

[8] Batini, C., Bolchini, D., Ceri, S., Matera, M., Maurino, A., & Paolin, P. (2007). The UM-MAIS methodology for multi-channel adaptive web information systems. *World Wide Web, 10(4),* 349-385.

[9] Bolchinie, D., & Paolini, P. (2004). Goal-driven requirements analysis for hypermedia-intensive web applications. *Requirements Engineering, 9(2),* 85-103.

[10] Comerio, M., De Paoli, F., Grega, S., Maurino, A., & Batini, C. (2007). WSMoD: A methodology for QoS-based web service design. *International Journal of Web Services Research, 4(2),* 33-60.

[11] Vdovjak, R., Frasincar, F., Houben, G. J., & Barna, P. (2003). Engineering semantic web information systems in Hera. *Journal of Web Engineering, 2(1-2),* 3-26.

[12] Garrigós Fernández, I. (2008). A-OOH: Extending web application design with dynamic personalization. PhD dissertation, University of Alicante, Spain.

[13] Garrigós, I., Mazón, J. N., & Trujillo, J. (2009). A requirements analysis approach for using i* in web engineering. *Proceedings of the International Conference on Web Engineering* (pp. 151-165).

[14] de Koch, N. P., Knapp, A., Zhang, G., & Baumeister, H. (2007). UML-based web engineering: An approach based on standards. In G. Rossi, O. Pastor, D. Schwabe, & L. Olsina (Eds.), *Web Engineering: Modelling and Implementing Web Applications* (pp. 157-191). London, UK: Springer.

[15] de Koch, N. P. (2001). software engineering for adaptive hypermedia systems. PhD dissertation, Ludwig Maximilian University of Munich.

[16] Baumeister, H., Knapp, A., de Koch, N. P., & Zhang, G. (2005). Modelling adaptivity with aspects. *Proceedings of the International Conference on Web Engineering* (pp. 406-416).

[17] Kitchenham, B., Linkman, S., & Law, D. (1997). DESMET: A methodology for evaluating software engineering methods and tools. *Computing and Contrological Engineering Journal, 8,* 120-126.

[18] Taromirad, M., & Ramsin, R. (2008). An appraisal of existing evaluation frameworks for agile methodologies. *Proceedings of the International IEEE Conference and Workshop on the Engineering of Computer Based Systems* (pp. 418-427).

[19] Ramsin, R., & Paige, R. F. (2010). Iterative criteria-based approach to engineering the requirements of software development methodologies. *IET Software, 4(2),* 91-104.

[20] Asadi, M., & Ramsin, R. (2008). MDA-based methodologies: An analytical survey. *Proceedings of the European Conference on Model Driven Architecture: Foundations and Applications* (pp. 419-431).

[21] Sadat, H., & Ghorbani, A. A. (2004). On the evaluation of adaptive web systems. *Proceedings of the Workshop on Web-Based Support Systems* (pp. 127-136).

[22] Boudaa, B., Camp, O., Hammoudi, S., & Chikh, M. A. (2012). Model-driven development of context-aware services: Issues, techniques and review. *Proceedings of the International IEEE Conference on Information Technology and e-Services* (pp. 1-8).

[23] Hussein, M., Han, J., & Colman, A. (2011). An approach to model-based development of context-aware

adaptive systems. *Proceedings of the IEEE Computer Software and Applications Conference* (pp. 205-214).

[24] Taylor, M. J., McWilliam, J., Forsyth, H., & Wade, S. (2002). Methodologies and website development: A survey of practice. *Information and Software Technology, 44(6),* 381-391.

[25] McDonald, A., & Welland, R. (2004). Evaluation of commercial web engineering processes. *Proceedings of the International Conference on Web Engineering* (pp. 167-170).

Mona Fadavi earned her BSc in computer engineering (software) from Khajeh Nasir Toosi University of Technology, Iran, in 2013. She is now a MSc student of computer engineering (software) at Sharif University of Technology. Her research project focuses on designing a model-driven approach for developing adaptive web systems.

Raman Ramsin earned his PhD in computer science from the University of York, UK, in 2006. He is an assistant professor at the Department of Computer Engineering, Sharif University of Technology. Dr. Ramsin is a professional member of ACM and IEEE, and his research focuses on object-oriented software engineering, situational method engineering, software patterns, and pattern-oriented development.