World Wide Web (2007) 10:349–385
DOI 10.1007/s11280-007-0025-x

The UM-MAIS Methodology for Multi-channel Adaptive Web Information Systems

C. Batini · D. Bolchini · S. Ceri · M. Matera · A. Maurino · P. Paolini

Received: 12 April 2006 / Revised: 29 January 2007 /
Accepted: 26 February 2007 / Published online: 5 May 2007

Abstract Multichannel Adaptive Web Information Systems (WISs) are emerging as a new class of information systems, characterized by their powerful use of mobility and context-awareness. Different methodologies have been proposed so far for the analysis and design of Multichannel Adaptive WISs, specifically focused on the front-end layer or the back-end layer, but no methodology has aimed to cover all the lifecycle and to design all the components that characterize Multichannel Adaptive WIS. This paper fills such a gap, by presenting UM-MAIS (Unified Methodology for Multichannel Adaptive Information Systems), a new methodology that capitalizes on well-established existing methods. It supports the analysis and design of the various components of Multichannel Adaptive WISs (including the user's experience) in a comprehensive and unified manner with special emphasis on context modeling, personalization, and adaptation.

Keywords adaptive web information systems · design methodologies · context modelling · requirements analysis · services oriented architecture · web applications

C. Batini · A. Maurino (✉)
Università di Milano Bicocca, Milano, Italy
e-mail: maurino@disco.unimib.it

C. Batini
e-mail: batini@disco.unimib.it

S. Ceri · M. Matera · P. Paolini
Politecnico di Milano, Milano, Italy

S. Ceri
e-mail: ceri@elet.polimi.it

M. Matera
e-mail: matera@elet.polimi.it

P. Paolini
e-mail: paolini@elet.polimi.it

D. Bolchini
Università della Svizzera Italiana, Lugano, Switzerland
e-mail: davide.bolchini@lu.unisi.ch

1 Introduction

The design and development of information systems have been often tackled from two autonomous and independent viewpoints. From one hand, significant efforts have been performed for the analysis, design, and implementation of the so called front-end layer, which is in charge of allowing end users to interact with information systems; the Web paradigm is the “de facto” standard to model the interaction between users and systems through dynamic hypertext interfaces. From the other hand, several studies have been conducted for supporting the design and development of the back-end layer, involving the definition of the operative and informative services that automate information systems. In this context, Service Oriented Architectures (SOAs) are assuming prominent relevance as a framework for (Web) service provisioning.

Recently, *Multichannel Adaptive Web Information Systems* (Multichannel Adaptive WISs) are emerging as a new class of information systems characterized by the powerful use of mobility and context-awareness. As traditional Web information systems, Multichannel Adaptive WISs consist of three main components: the data source, the hypertext front-end and the back-end services. So far the design of Multichannel Adaptive WISs has been undertaken by methods that focus separately on specific components and phases of the lifecycle: several methods cover requirement analysis [9, 11, 12, 39], other methods cover conceptual modeling [16, 17, 26, 44], other methods address service design [7, 37]. None of them is sufficient for managing the complexity introduced by mobility and context-awareness. In particular:

– Methods are unable to guide analysts and designers in discovering and implementing requirements, which are specific to mobility and context-awareness.
– Even when providing conceptual abstractions for capturing adaptive behaviors, methods do not address the design of the back-end layer and of the interaction with external services—at most they provide constructs for representing the invocation of “black-box” services [34].
– When methods support service adaptivity, they do not focus on the front-end layer or on the non-functional requirements that drive service customization [21].

Therefore we believe that new methods, techniques and tools are required in order to approach the peculiarity of the adaptability and multi-channel requirements along all the dimensions. First of all, adaptability issues must be analyzed and then implemented by adopting “ad-hoc” techniques. Also, while separation of concerns is in general advantageous for reducing the effort of system development, we believe that “communication” between the different design dimensions is required. Therefore, separate methods addressing different dimensions can be fruitfully adopted only if cross-interactions and possible “adaptors” are clearly defined.

Given the previous premises, this paper proposes a new methodology, UM-MAIS (Unified Methodology for Multichannel Adaptive Information Systems), that has been conceived in the context of the MAIS Italian project [42]. UM-MAIS results from the integration of three existing and field-tested methods for requirement analysis [12], hypertext front-end design [16] and service design [21]. UM-MAIS capitalizes on the strength of these three methods, and tries to overcome the limitations of the current approaches by:

– Offering a set of analysis and design techniques that specifically focus on adaptivity and multi-channel issues.
– Covering all the main dimensions characterizing Multichannel Adaptive WISs, by means of a unified method and a pool of tools assisting the whole development cycle.

– Offering a systematic view over the Multichannel Adaptive WIS development, clarifying the different analysis and design activities, the flow of knowledge and the cross-interactions between the different figures undertaking such activities.

The resulting framework supports an agile paradigm [1] that allows designers to refine the different Multichannel Adaptive WIS components through a number of successive iterations. It is also in line with the model-driven approach proposed by OMG [46], since it proposes the definition of a set of abstract models that are then translated and/or refined into new model representations, as long as the implementation of the application code is generated.

This paper illustrates the set of methods, techniques and tools characterizing UM-MAIS, and also shows the methodology at work for the development of a Multichannel Adaptive WIS for the management of Italian archaeological sites. In particular, Section 2 discusses the rationale and the framework of our research, by illustrating the main motivations behind the definition of UM-MAIS, and comparing our approach with related work. Section 3 outlines the methodology, by introducing a global view of the main phases and activities, and illustrates the running case that we use throughout the paper for exemplifying concepts. The following sections then provide details about how adaptability and multi-channel issues are approached in the different design phases. In particular, Section 4 describes the requirement management activities. Section 5 illustrates the activities in the High-Level Design phases, which allow designers to sketch an overall picture of the Multichannel Adaptive WIS, by defining its "in-the-large organization" for data, hypertext, and service design. Sections 6 and 7 deal, respectively, with Low-Level Hypertext and Service Design. Section 8 then illustrates the tools developed so far for supporting the activities previously described. Finally, Section 9 draws our conclusions and future work.

2 Rationale and background

The characterizing feature of Multichannel Adaptive WISs is the ability to capture and exploit the execution context. Context can be captured at several architectural levels, ranging from the application front-end, capturing context by means of mobile devices, to the application back-end, capturing context-specific state information. Figure 1 represents the context reference model that has been conceived within the MAIS project [42], and that we also adopt in this paper. According to Figure 1 the context is composed by the *user profile*, the *environment*, and the *channel*.

- The user profile describes the properties associated with the user; it is distinguished in *domain-independent* and *domain-dependent* user profile. The former contains all the characteristics that describe the users along their social distinctive properties. It includes personal data, their physical abilities and general interests. The latter refers to the specific domain in which the context is considered, and it usually contains preferences related to specific services.
- The environment collects the information about the geographical position, the ambient condition, the temporal details and the actions that characterize the interaction of the users with the surrounding space.
- Finally, the channel describes the elements that characterize the interaction of the users with the platform used to access services. The considered model characterizes the channel through its device, network, application protocol and network interface features.

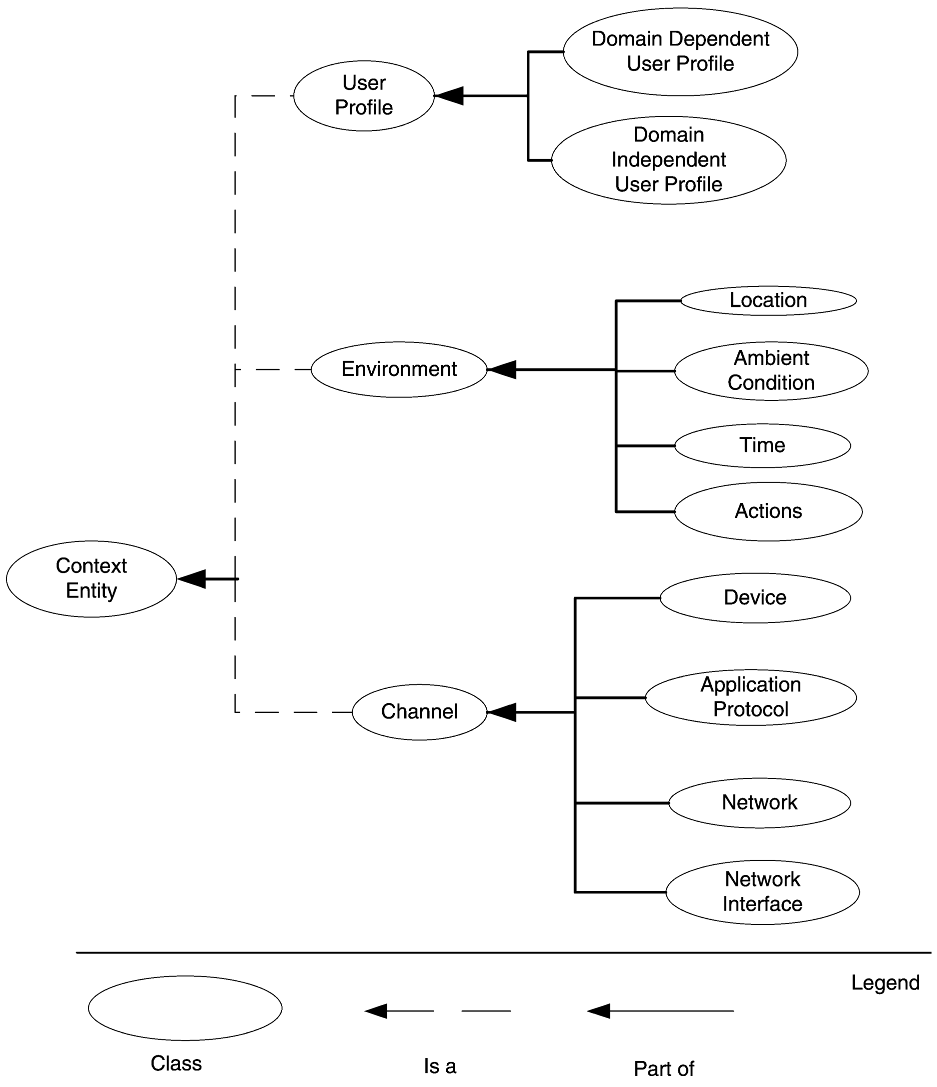

Figure 1 The context model [14].

At the front-end layer, the emphasis is on adaptability with respect to the user device, the user profile, and the context of use. The main goal is to provide users with contents adhering to the format and the presentation modality that are more appropriate for the current context of use, and with services for which quality (e.g., availability, connectivity, quality of data, etc.) is dynamically negotiated.

At the back-end layer, the emphasis is on the adaptability with respect to the user profile, the environment, and the channel. The back-end infrastructure should therefore enable the provision of services which best fit with the service requests and expected quality of service (QoS). Moreover, services have to be adapted to the specific communication channel being used (for example by considering peculiar network speed, or the device capabilities).

So far the design of the different architectural components addressing the previous issues has been approached by several and separate methods, each one focusing on a specific architectural layer. The methodology that we propose in this paper proposes the seamless integration of three individual methods originally focusing on specific issues, with the aim of covering all the different components of a Multichannel Adaptive WIS, as well as all the phases of its lifecycle. In particular:

- *Requirement management*, i.e. the phase of the method initiating the lifecycle at the highest level of abstraction, is based on the AWARE method (Analysis of Web Application REquirements) [12]. In line with the traditional approach of goal-oriented requirement engineering [3, 22, 38], AWARE recognizes the central role of all the relevant stakeholders and their goals in eliciting, analyzing and specifying the requirements for a Web application. AWARE also assumes that the "why" of the stakeholders' requirements has to be sought and documented, in order to keep track of the reasons behind the requirements and the design decisions.
- *Web application design* (both data and hypertext design) is based on the WebML method, a visual language combined with a methodology that supports the specification of the content structure of a Web application and the organization of the hypertext interfaces for content presentation and service invocation [16, 34]. The WebML-based development is also supported by a case tool that assists the visual modeling of the application schemas and their automatic transformation into the application code, executed by means of a proper runtime environment.
- *Service design* is based on the WSMoD (Web Service Modeling and Design) methodology [21], which addresses the design of Web services according to specific quality of services (QoSs) rather than functional descriptions only. Furthermore, WSMoD exploits general knowledge available, expressed by ontologies describing services, their qualities, and the context of use, to help the designer in expressing service requirements in terms of design artifacts. WSMoD improves the effectiveness of the process, defining a Platform Independent Model that includes the description of specific context of service provision, without considering implementation details.

The three stand-alone methods have already proved their effectiveness in several experiences. The UM-MAIS methodology combines them trying to cover all the ground from the high-level user requirements down to the deployment of Web application and services. In particular, the emphasis is on: (1) specializing the methods for application analysis and design to cope with adaptivity and multi-channel issues [20, 50], as well as (2) coupling the method in a loose manner, as it has been envisioned and tested in the context of the MAIS project [42]. The overall goal is to provide a comprehensive framework, assisting the whole development cycle through a precise flow of knowledge between the different analysis and design activities, by preserving *separation of concerns* through the adoption of each individual method to effectively manage some specific aspects.

In order to compare our methodology with other relevant works proposed so far, in the rest of this section we will illustrate some approaches in the three fields of action of requirement engineering, Web application design and service design. The described related works further highlight the lack of unified methodologies.

2.1 Related work

In the research area of requirement engineering, one of the most consolidated paradigms is the so-called Goal-Oriented Requirement Engineering (GORE) [53]. The methods relying on this paradigm (see for example KAOS (Knowledge Acquisition in automated

Specification) [57], GBRAM (Goal-Based Requirement Analysis Method) [3], i* [56]) all share a common assumption: before specifying the requirements of a system it is crucial to elicit and identify the goals of all the relevant stakeholders, since they are the key drivers of the whole development. Existing goal-oriented approaches focus mainly on traditional information systems, mission-critical applications or complex software systems in general, with the aim of supporting a clear and consistent specification of the operations and transactions to be performed by the system. However, the domain of Web, which is characterized by information- and communication-intensive applications that need to convey structured contents and messages to a variety of target audiences, has received little attention from the requirement engineering community. By addressing this lack, the UM-MAIS methodology provides the analysts with a model, called AWARE [12], consisting of a simple set of concepts, terms, and notations to carry out requirement analysis in this realm. In particular, AWARE indicates how to elaborate the different user profiles by representing the target audience of the site. It takes their characteristics into account, as well as their goals, tasks and expectations with respect to the application-to-be.

Several approaches have been proposed for the design and development of Web applications, addressing advanced adaptation and personalization mechanisms [23, 26, 31, 44]. Recent research efforts specifically address the special needs of mobile Web applications and portable device characteristics [30]. However, despite the numerous advantages offered by conceptual, model-based development of Web applications [25], only few attempts exists that aim to model context-aware behaviours at a conceptual level.

On top of the model-based framework of the Hera project, authors propose a component-based XML document format for the implementation and deployment of component-based, adaptive Web presentations (AMACONT project), with particular emphasis on the context of use [5, 24]. Adaptation depends on user profile data and device characteristics, and it mainly concerns layout and presentation properties of Web pages. The implementation of AMACONT-based applications is supported by an automatic code generation mechanism for adaptive documents and multiple communication channels, starting from AMACONT components and Hera schemas. However, the Hera methodology adopted for specifying the application schemas is based on a conventional, non-adaptive interpretation of hypertext. Barna et al. [5] address exactly the above mentioned lack of dynamism at the model level, and show how the Hera design methodology [54] can be used successfully for the design of adaptive Web applications. By appropriately updating the user model at runtime, Hera schema views defined over the application data can yield adaptive characteristics.

SiteLang [49] is an algebraic language with an operational semantics based on entity-relationship structure and abstract state machines. It allows the specification of entire Web sites, i.e. of structure (data schema, static integrity constraints), behavior (processes and dynamic semantics), information support (views, units and containers), and interaction (scenes, media objects and dialogue steps), based on the so-called application “stories” [8]. The modeling approach takes a variety of context types (Customer’s scenario context, Vendor’s WIS-context, Developer’s WIS-context, and WIS’s scene context) into account. Nevertheless, this method has the limit of making the implementation less intuitive with respect to the model-driven approaches mentioned above.

Among the other proposals, Baumeister et al. [6] explore *Aspect-Oriented Programming* techniques for modeling adaptability in the context of the UML-based Web Engineering method (UWE [31]). The main contribution of this work can be identified in the strong separation of concerns between navigation and adaptation, achieved by interpreting adaptation as an orthogonal aspect with respect to application modeling. This assumption is in line with our approach that considers adaptability as an orthogonal design dimension, which considers context as the primary actor influencing the application and the service behaviour. However, in this approach the authors concentrate mainly on some limited aspects for front-end adaptability, such as link hiding, adaptive link annotation and adaptive link generation, while content and service adaptation is not considered.

In UM-MAIS we propose the use of a conceptual model, WebML [16] for Web application design, trying to strengthen the advantages offered by the above mentioned methods and reduce some of their lacks. Our approach is based on a model-based paradigm, which offers high-level intuitive constructs for specifying the application structure and behaviour. Due to the formal definition of construct semantics, the resulting specifications can then be transformed into running code, by means of consolidated generative techniques [18], thus facilitating the overall development process.

Different approaches (such as agent-based and model driven) for the design of Web services have been proposed by many research and industrial projects. The works presented in [29, 47, 48] exploit the use of an MDA approach for designing Web services. In [48] the authors present an example of a tool for both modeling a business project by means of a Platform Independent Model and by generating the corresponding WSDL description. [47], while in [29] the authors present an MDA strategy to develop Web services. As in our methodology, a Platform Independent Model to model Web services is produced, and then, WSDL and Ws-BPEL specifications are produced by means of a translator tool. However, these methodologies do not provide Web service descriptions with QoS specifications and do not use ontologies to support designers. A different approach is shown in [37] where the authors propose agent-oriented software development techniques to design Web services. These techniques support early and late requirement analysis, as well as architectural and detailed design, but do not offer ontology support for QoS description.

Recently, Self-serv [7], another model-driven approach for designing Web services, has been presented to facilitate the rapid and scalable composition of Web services. It supports the declarative composition of new services from existing ones, the composition of Web services with multi attributes, dynamic selection, and peer-to-peer orchestration of composite service executions. Self-serv addresses QoS issues, defines a fixed Web services quality model, composed of five different QoS dimensions, namely execution price, execution duration, reputation, reliability, and availability.

Despite all the previous methods focusing on specific aspects, the complexity of Multichannel Adaptive WISs demands for integrated methods, able to cover all the facets of the layered design, and for the identification and definition of possible cross-interactions. In our approach, in order to facilitate the transition between high-level goals to Web site requirements, in the requirement management phase a goal refinement technique and hypermedia taxonomy to organize the requirements set are adopted. Furthermore, while all the previous methods for Web application design neglect the issues related to the design of the back-end layer, our notation for Web application modelling also includes some primitives for service invocation [34]; the methodology then has the advantage to combine application design with service design by addressing the cross interactions and the interfaces between the two design activities, as well as the occurring flow of knowledge.

3 The UM-MAIS methodology at a glance

Figure 2 shows the main phases of the methodology. UM-MAIS starts with *Requirement Management*, a phase resulting from a suitable enrichment of the AWARE methodology [12]. Inputs for this phase are the user-centered business requirements collected during the

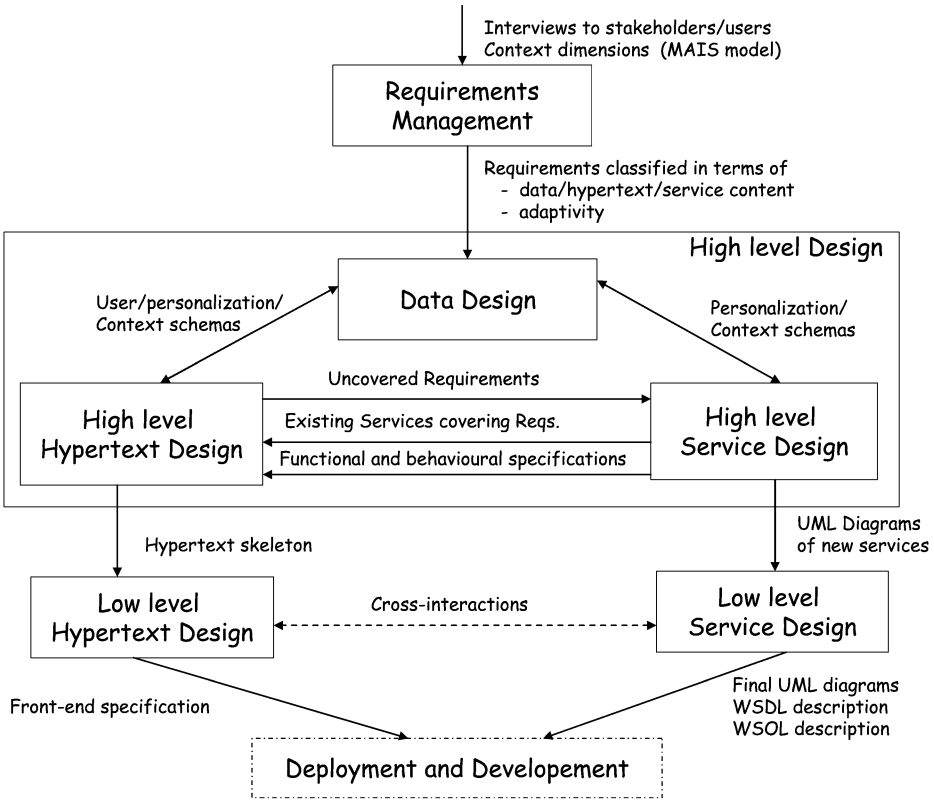
Interviews to stakeholders/users
Context dimensions (MAIS model)
Requirements Management
Requirements classified in terms of
- data/hypertext/service content
- adaptivity
High level Design
Data Design
User/personalization/ Context schemas
Personalization/ Context schemas
Uncovered Requirements
High level Hypertext Design
Existing Services covering Reqs.
Functional and behavioural specifications
High level Service Design
Hypertext skeleton
UML Diagrams of new services
Low level Hypertext Design
Cross-interactions
Low level Service Design
Front-end specification
Final UML diagrams
WSDL description
WSOL description
Deployment and Developement

Figure 2 Phases of the UM-MAIS methodology.

initial customer interviews. Requirement analysis enables the capture of the picture of the involved project stakeholders (including users), their (communication, business and operational) goals and salient scenarios of use that can highlight adaptability needs. Finally, an organized set of requirements is provided—concerning various application aspects, from data, to hypertext, to services—as important indications to be considered by the designers.

The output of the requirement analysis is the input to the Design phases, namely the High-Level Design and Low-Level Design. Main design steps in UM-MAIS refer to:

1. *Data design*, which produces a first representation of the information needed by the hypertext front-end and the invoked services. It drives the overall application development and it is the first contact point between front-end (hypertext) and back-end (service) design. During *Data design*, the main information assets and their relationships are specified by using well-known notations, such as E-R schemas or UML class diagrams.
2. *High-Level Hypertext Design* and *Service Design*, which are strictly related and clarify "in-the-large" the organization of the hypertext front-end and the main functionalities to be covered by back-end and external services. They define a preliminary coarse structure of the front-end, by identifying the hypertexts by addressing the functionality of the application users, and their organization throughout different areas and pages. This step is based on the use of the WebML model and method [16], and pays special attention to identifying adaptive and context-aware behaviors. Moreover, during this phase, existing services supporting the application are discovered, and new services

required for the specific needs of the application are specified. This task is based on the use of WSMoD methodology [21].
3. *Low-Level Hypertext Design* and *Service Design*, which detail and formalize the above specifications, and can be performed in parallel, with loose interaction. Also in these phases both WebML and WSMoD methods are adopted.

Some interactions between High-Level Hypertext and Service Design must occur. First of all, the hypertext designer highlights some activities that cannot be fully addressed through the hypertext design and that therefore need the invocation of external (e-services) operations. Such uncovered requirements are checked by the service designer against existing services; if some suitable services already exist, then the service designer provides the hypertext designers with the details (for example a WSDL specification) about how to invoke them within the application. Conversely, the service designer provides the hypertext designer with a high-level specification of the service(s) to be developed, including their functional and non functional requirements. This information is useful to identify the flow of information (i.e., input and output parameters) from the hypertext modules (e.g., pages) invoking the service(s) to the service(s) and vice-versa. It is worth noting that the previous cross-interactions, in turn may also suggest new data to be included in the overall data schema previously defined. Therefore, an iterative flow characterizes the High-Level Design phase.

The Low-Level Hypertext and Service Design phases can then be performed almost independently, still adopting WebML and WSMoD to respectively produce, a detailed specification of the hypertext front-end and of back-end services. This specification, when provides the input to the implementation phase, consists of:

– A WebML-based, detailed hypertext specification, which describes the organization of contents in the hypertext pages, possible invocation of back-end services, and the adaptability rules to be executed to adapt the hypertext front-end (i.e., navigation and page contents and presentation).
– A detailed specification of services in terms of UML diagrams (i.e., class, sequence and activity diagrams) and standard languages for service description, namely WSDL to describe the functional interface of services, and WSOL (Web Services Offering Language) [51] to describe non functional requirements that a service provides.

In the Low-Level Design phase, some feedback actions over data design are still possible, as the detailed description of hypertexts and services might highlight some requirements which did not emerge before. Due to the possible new requirements, it is possible that some fragments defined in the common data schema and used both by the front-end and the back-end layer need to be modified. In this case, all the choices in the two layers related to these fragments need to be re-discussed. This leads to a number of iterations and cross-interactions between the designs of the two layers.

It is worth noting that UM-MAIS does not impose a rigid progression of the design activities; rather it suggests a cycle of cross-refinements, in which the three activities (data design, front-end design and back-end design) cooperate. The mutual interactions may depend on a technological, organization and legal constraints, such as:

– The availability of legacy systems to be integrated in the new design (services may be already available; thus their detailed specification is already available and gives precise constraints to the data design activity from the first iterations of design).
– The internal business processes already existing, and the changes that the new WIS imposes upon the current organization of activities.

– Some legal restrictions, posing constraints over the services to be adopted (e.g., digital signature for electronic document exchange).
– General directives (e.g., accessibility requirements) constraining the front-end organization, independently from the invoked back-end services.

3.1 Running case

In the following sections, we will describe in more details the methodological steps and the suggested activities for Requirement Management, High-Level Design and Low-Level Design. We will also exemplify the methodology by discussing the design of a Multichannel Adaptive WIS for the management of Italian archaeological sites. The system is in charge of extending an existing project MARIS [15] adding functionalities for the acquisition of data on cultural heritages.

The broad presence of sites and archeological items over the Italian territory requests the use of mobile devices and networks to automate the data acquisition process. The mission of the system is to define a complete and up-to-date repository of the state of all the cultural heritages with the goal of building the Italian risk map of archaeological sites. In particular, the risk map has to allow users to process data regarding territorial danger factors and monuments' vulnerability conditions. Special attention has to be put in the data acquisition phase.

Therefore, each site must be completely described, by filling in a site description card. The data acquisition is realized by operative teams that, by means of mobile devices such as laptops, PDAs or smart phones, they can collect accurate information in an electronic way. If, during the data acquisition phase, there is clear evidence of an immediate risk for the monument, Field Operators have to immediately notify the headquarters about the danger. During archeological campaigns, operative teams are placed in mobile camps that communicate with the headquarters by means of internet connections. Therefore, different users exploit the information system in different ways, according to their preferences and goals. In particular:

– *Headquarter Operators* need to interact with the information system to plan the reconnaissance campaigns.
– *Field Operators* need to access the site description card for the site currently visited. Among them, a *field manager* may coordinate the work of the other operators.
– *Mobile Camp Operators* need to know the geographic position of operative teams, their features and competencies in order to manage the reconnaissance campaign, for example by moving an operative team from an archaeological site to another one when some specific interventions are required.
– *Local and state administrators* need to access the information system to improve the decision-making processes for conservative interventions.

By discussing the previous scenarios in the rest of the paper we will show how the UM-MAIS methodology can provide support in covering the analysis and the design of a Multichannel Adaptive WIS in a unified manner. For more information about the system, the interested reader is referred to [35, 36], and http://www.mais-project.it/html/riskmanagement.htm.

4 Requirement management

As shown in Figure 3 requirement management proceeds along two activities: *Requirement elicitation* and *Requirement analysis and classification*. These two activities are

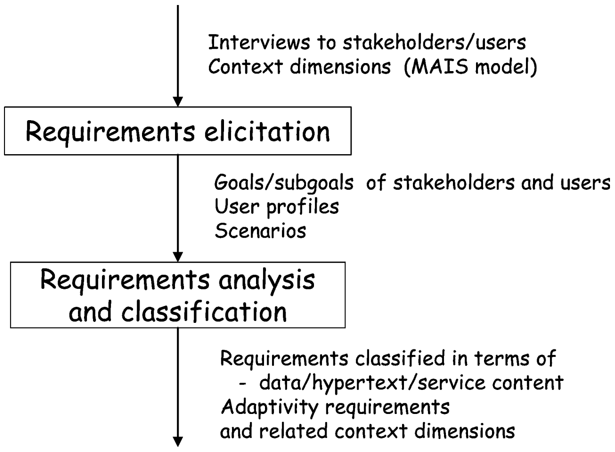
Interviews to stakeholders/users
Context dimensions (MAIS model)
Requirements elicitation
Goals/subgoals of stakeholders and users
User profiles
Scenarios
Requirements analysis and classification
Requirements classified in terms of
- data/hypertext/service content
Adaptivity requirements
and related context dimensions

Figure 3 Steps in the requirement analysis phase.

accomplished in accordance with the AWARE methodology. In order to cover the design of Multi-channel Adaptive WISs, in the rest of this sections we will show how AWARE has been enriched to address requirements related to adaptivity and to Web service integration.

4.1 Requirement elicitation

Requirement elicitation is the activity aiming to make requirements surface. As suggested by goal-based approaches to requirements analysis, requirements are not just "there" to be found, but they may emerge after the identification of the relevant users involved in the project and their goals.

The elicitation process is facilitated by a set of guiding principles, which support a complex and difficult activity, often taking much of the initial resources of the project, and crucial for the strategic decisions to be taken afterwards. Two of the most important users have to be considered: *stakeholders* and *final users*. The former are those people who have an interest in the success of the application or may share knowledge and visions for its success. In our case study, a stakeholder playing the role of "financial partner" or main funder of the initiative is the Municipality, together with other actors, such as the private associations and public institutions investing for the preservation of the cultural heritage sites at issue. *Final users* are also an important category of stakeholders. They will use the services provided by the system, and services have to be adapted to their characteristics. In this phase, users may be described in terms of a *"personal" characteristic* of an archetypal visitor. A *user profile* aggregates one or more relevant characteristics, which tentatively describe a potential visitor of the application.

In our running example, relevant user profiles are defined on the basis of their technical role, they are: *Field Operators, Headquarter operators, Mobile Camp Operators, Local and state administrations*. Both stakeholders and users, depending on their specific roles, and own goals with respect to the application to build. More specifically:

1. Stakeholders have a direct interest in the successful deployment and use (e.g. the client or his representatives) of the applications. Other stakeholders may not have goals, but they can project their *visions* on the application. The municipality, for example, aims to give the highest visibility to the project on regional newspapers and communication channels.

2. The users have *goals*, which should be defined as plausible motivations for visiting the application, or the objectives of their interaction. User goals may vary in granularity from low-level, specific and punctual information seeking ("find the exact position of the Roman tomb in the park"), also called—functional goals—to higher-level, open-ended, ill-defined needs or expectations ("decide if the archaeological site is worth evaluating"), also called soft-goals [56].

Typical questions to identify that may help to elicit the goals of the users are:

- Why should a given user make use of the application?
- What is the expected outcome of his/her interaction?
- In which circumstances should s/he use the application? For which purpose?

The raw material emerged during the initial elicitation has to be refined in order to more precisely represent the concrete goals and needs of users, and to be usable by the analysts, and fed into design. AWARE adopts a refinement process to pass from high-level goals of all stakeholders (including users) to sub-goals and eventually to application requirements. The analysis of such material may be guided by the following lines of inquiry:

- *Clarify the meaning of a high-level goal*—Often user goals and main stakeholder goals are too vague, abstract or generic, posing obstacles to devising operative indications for the designers. For example, if a user goal is "Update the level of knowledge," analysts should ask: what does it mean specifically? How can it be specified?
- *Refine into subgoals*—Analysts should elicit possible subgoals which may contribute to the accomplishments the long-term goal. For example, "How can the checking of the site status be accomplished?." Plausible subgoals are: "to update description cards," or "to orchestrate the team work."

Organizing the material gathered during elicitation more systematic way, a table for each user profile may be sketched, like the one illustrated in the following.

Profile name: *Field operator*
Goals:
• Update the level of knowledge of the site
• Check the site status and report to the headquarter
Sub goals:
• Update site description cards
• If field manager, orchestrate the team work
• Manage emergency in situ

An effective activity to facilitate this elicitation process is the development of user scenarios. *Scenarios* are task-oriented vivid descriptions of envisioned use of the application, which can assist analysts in discovering new requirements, exemplify goals, surface new goals and better define stakeholders. Scenarios are commonly recognized as powerful drivers for goal-based approaches; they are partial descriptions of the wide spectrum of the potential interaction capabilities of the application to be developed. Therefore, they should be used either to exemplify and make more concrete elements of the goal analysis or to anticipate design details and interaction paradigm already during requirements. An example of the scenario for the Field Operators is shown in the following.

Scenario 1: *Campaign reschedule*
Mobile Camp Operators need to cover the analysis of a new site, for which more hotspots than expected have just been discovered. To this end, Mobile Camp Operators need to know which operative teams are active and how many Field Operators are involved, as well as where the various operative teams are located, in order to select the most appropriate one with respect to the kind of required intervention.
Through the systems the Mobile Camp Operators locate two operative teams with a competence profile matching the required task, select both teams and send the intervention request to the field managers with task details (where to go, what to do, etc.). The field manager receives the message and, according to the network conditions available, sends the task details to their team members located nearby on the site and ask them all to reach him at the mobile camp to organize the intervention. Each team member reads the requested intervention and saves the results of the current activity. Once the mobile camp is reached, one of the teams moves to the new location and starts the campaign; the field manager sends two Field Operators to scan the site to the specific hotspot indicated by the headquarter, in order to create the descriptive cards for all the hotspots identified.

4.2 Requirement analysis and classification

During requirement analysis and classification, on the basis of the input received by the elicitation activity, and thanks to the concrete examples of uses depicted in the scenarios, an organized set of requirements is defined. Requirements are expressed at a high-level of abstraction, in order to capture the key aspects of the system to be developed. In real projects, the definition of requirements is iterative, in such a way that requirements may be detailed, better specified or changed according to the development of the design. In order to relate better the requirement management activity with the design phases, requirements are now classified in terms of data, hypertext/application, and service requirements, according to the prevailing issue addressed.

Data Requirements describe the information assets that the application should manage to accomplish the expected goals. An important dimension of data requirements are the content requirements, meant as the set of ideas and messages that the site communicates to its users. Ideas and messages are mainly specified in term of provided information chunks. Examples of of this kind of requirements for the case study are the following:

– *DR1 Site information.* The application should contain detailed information about all the archeological sites managed by the system. For each archeological site, detailed information about its location and its preservation status of the relevant hotspots is provided.
– *DR2 Team information*. The application should also contain information about the team work status, their location, as well as about who Field Operators of a given team are, what their skills and tasks are.

Hypertext Requirements describe the strategies for the users to access contents through the hypertext front-end. An important dimension of the hypertext requirements concern the access paths to the content, meant as the indications about the navigational paths available to the user in order to reach the needed content. Examples of requirements of this kind for the case study are the following:

– *HR1 Access to sites*. Archeological sites may be accessed by location, size, archeological importance, preservation status
– *HR2 Access to teams.* The access to the teams is organized by current GPS position, and by team skills and is shown according to the device of final users

Service Requirements concern service request and delivery. They should assume that the design of Web services cannot be confined to a technological problem, since it has deep business, social, and organizational impacts. As a consequence, their design must take into account all aspects related to the context of application. An important dimension of this class of requirements is the one concerning the operations that are visible to users to complete a task. User operation requirements especially influence the organization of the hypertext front-end. Examples of requirements of this kind for the case study are the following:

- *SR1 Manage description card.* Field Operators can create and edit the description card of each hotspot
- *SR2 Field Operators communicate with mobile camp.* Field Operators can send the updated information to the Mobile Camp Operators according to the specific features of the mobile network used by Field Operators. Mobile Camp Operators can call any field operator when needed and provide instructions on tasks
- *SR3 Emergency communication.* Field Operators indicate information about any emergency detected on site to the mobile camp by using the fastest mobile channel available.

Once a first list of requirements is founded, the Requirement analysis and classification step consist in evaluating the impact of the adaptability on the application. *Adaptability requirements* can be defined by analyzing the impact on some of the context dimensions (e.g. user profile, location, action, channel-network and channel-device) on the requirements elicited and classified so far. Some examples of questions in which adaptability issues might surface are:

User profile

- Which aspects of the application are common to all user profiles?
- Where does the application mostly differ for one user profile with respect to another? How?
- Which physical conditions can reduce usability?

Location

- How does a change in location affect the interaction with the application? For which users?
- In which circumstances should the application adapt to the user's location?
- What elements of the location (light, temperature, distance from a point, ...) are relevant for such adaptive behavior?

Action

- Which user activity provokes a modification in the application behavior?
- Which user profiles will carry it out?
- What elements of the application are typically affected?

Channel-network

- What is the expected quality of service offered by the network links connecting the various locations where the application will be deployed (from headquarter to mobile camp, from mobile camp to field operator, and vice versa)?
- What kind of networks can be considered?
- What main characteristics differentiate networks from one another?

Table 1 Adaptability requirements resulting from the cross checking of context dimensions.

Context dimensions/ requirements	User profile	Location	Action	Channel–network	Channel–device
DR1–site information			–	CR1–site information are available to the field operator according to the network conditions available	CR6/ CR7
DR2–team information			–		CR6/ CR7
HR1–access to sites	CR2–headquarter and mobile camp view all sites under coordination; field operator views only the site s/he is operating with		–		CR6/ CR7
HR2–access to teams	CR3–headquarter and mobile camp view all sites under coordination; field operator views only the site s/he is operating with		–		CR6/ CR7
SR1–manage description cards		CR4–the system proactively indicates to the Filed operator the information about the hotspot s/he is close to	–		CR6/ CR7
SR2–field operators communicate with mobile camp			–	CR5–the most reliable and efficient network connection is selected according to the available service	CR6/ CR7
SR3–emergency communication			–		CR6/ CR7

Channel-device

- What are the devices typically used by the user profiles?
- In which circumstances will a given user profile make use of his/her device?
- What main characteristics differentiate devices from one another?

A matrix (as the one shown in Table 1) can be used to understand what elements of the application are context-sensitive and should give input for designing an adaptive behavior. Each intersection between a requirement and a context dimension provokes reflection on the kind of adaptation desired and provides more analytic indications that will be considered during design. Furthermore, new requirements are discovered that enrich the requirement set.

As shown in the matrix, some adaptability requirements are cross-concerns, and thus valid for the application in its whole. They are:

– *CR6 Multi-device access.* The application should be made available on PDAs (for Field Operators), laptops (for the Mobile Camp Operators), as well as on traditional desktops (for Headquarter operators and Local and State administrations).
– *CR7 Device-adaptive interaction.* The application should be delivered and browsed in a format appropriate to the current adopted devices and also the current activity they are performing.

Although a requirement may concern more than one dimension, UM-MAIS suggests that it is better to refine a requirement to the point where exactly one dimension can be assigned to it. If a requirement cannot be easily and clearly assigned to exactly one dimension, then it is still too general to serve as input for design and should be further refined. This separation of concerns facilitates the achievement of an agreed granularity level. Not all dimensions have to be used in any project. Application requirements for a specific project may be specified only on some dimensions, which are considered relevant and meaningful at requirement time.

Once the set of requirements has been defined, it is important to perform a *traceability* activity [10, 27], i.e. to pass smoothly towards design by highlighting the relationships between the defined overall goals, the requirements gathered, and the subsequent design artifacts. In general, we can consider a specification as "traceable" if the origin of each of the artifacts or objects described in such a specification is clear and if it facilitates the referencing of each object in future development or enhancement documentation. In UM-MAIS, we report traceability in two directions, namely *backward traceability* (from requirements to the goals) and *forward traceability* (from requirements to high-level design artifacts), e.g., as to backward traceability, requirements should be connected to the goals in order to see how the defined requirements satisfy or address the goals. Moreover, it should be clear how the goals have been taken into consideration in the requirements definition.

5 High-level design

In this phase the requirements previously collected are translated into specifications, clarifying the organization of the data schema, of the hypertext front-end and of the services.

5.1 Data design

UM-MAIS is data-driven: the design starts from the specification of the main information assets; it then proceeds with the definition of the front-end able to support both the hypertext, and with the broad definition of the needed services.

Data design covers the translation into information objects of knowledge about the front-end and service domain. For adaptive context-aware systems, the data-driven approach primarily implies the representation at the data level of context, which in our case includes user profile, and distribution channels. The data designer starts from the requirements defined in the previous phase and selects entities that will be involved in the specific application, from the context reference model presented in [14] (see Figure 1). In this way, both content customization and context-based adaptive behaviors for the hypertext front-end can be achieved in a totally dynamic way.

The requirements about content and service are used for identifying and organizing the main application data around which the application and the invoked services are built. Requirements about the user profile and the channel model instead are translated into three different "sub-schemas" complementing the application data:

– *User sub-schema* consisting of some information entities representing user profiles and possible users' clustering into groups. User clustering may depend on user common features, or simply on the role played by users within the application.
– *Personalization sub-schema*, consisting of entities and relationships that interconnect users with application objects (i.e., entities in the application sub-schema), thus expressing both user preferences and ownership over some data instances. In general, relationships defined between users and any other data allow personalizing thus data, with respect to the identified types of users.
– *Context sub-schema*, which includes information objects, such as *Device, Location, Channel* and *Activity*, describes particular properties of context to be considered by the application front-end and by services for providing adaptability. Associations between context data objects and user objects represent the connection of each user to her/his (personal) context. The context sub-schema can vary depending on the front-end and service domain and, also, on the adaptability goals to be fulfilled. Therefore, our approach only prescribes to have the user as the center of this sub-schema, since these entities represents the actual starting point for navigating the context model and extracting context information.

Figure 4 illustrates the data schema for our running application; it represents the most relevant information entities that derive from the requirements illustrated in Section 3, as well as the meta-data required for managing adaptability and context-awareness. In particular:

– The *User sub-schema* is centered on the entity **User**, which provides a basic profile of the users pf the application. The entity **Group** represents clusters of users, and supports the management of access rights for groups of users. As better described in Section 5.2,

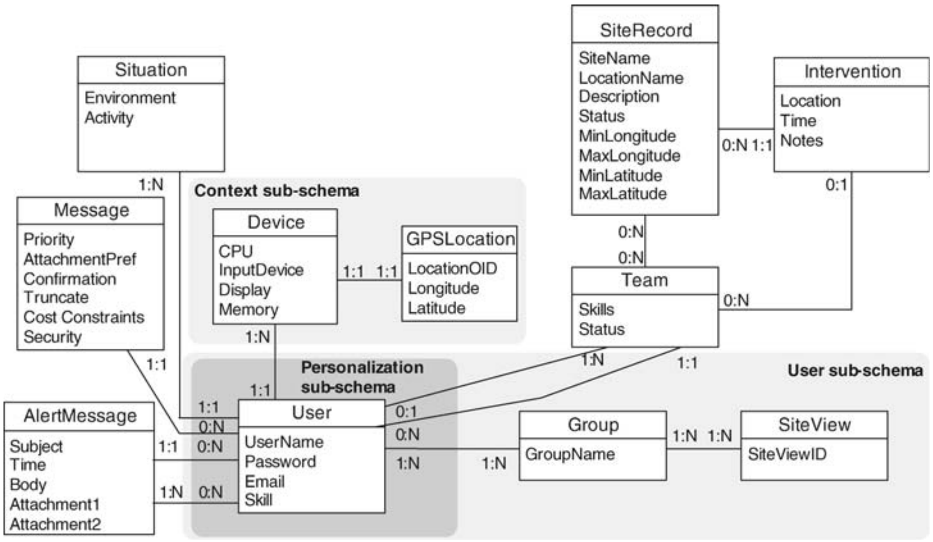

Figure 4 High-level data schema for the case study.

the entity **Site View** allows associating users (and user groups) with views over the application's data source. In the association of the case of adaptive context-aware applications, users may indeed require different interaction and navigation structures, according to varying context properties and different devices.

– The *Personalization sub-schema* includes all the relationships departing from the entity User, which express both user preferences and ownership over the instances of the target entities. For example, the relationships between **User** and **AlertMessage** represent the association of the users with the Alert messages sent and/or received. In general relationships defined between the entity **User** and any other entity allows the personalization of the entity contents with respect to the identity of users.
– *Context sub-schema*, which includes some entities, such as **Device** and **GPSLocation**, describes particular properties of context to be considered by the application which provides adaptability. Context entities are connected to the entity User to associate each user to her/his (personal) context.

The other entities in the schema represent application data. They store the archeological sites to be monitored, the interventions that operative teams are required to execute over archeological sites, and the alert messages that the Mobile Camp Operators may send to the operative teams to require exceptional interventions. Moreover, as result of the scenario of Section 4, a set of data are needed to represent the message notifications to Field Operators sent by Mobile Camp Operators.

5.2 High-level hypertext design

For the design of the application front-end, we capitalize on the experience gained with the WebML methodology [16]. WebML proposes some high-level abstractions characterized by the following properties:

1. The Web application provides its users with one or more *site views*, i.e., one or more hypertexts, each one serving the needs of a specific device, or of a specific set of functions supporting a specific user profile and activity. Every site view is divided into *areas*, and each area contains a set of *pages*, which are the actual interface elements delivered to the users. Pages consist of units, conceptually representing content elements to be published under different formats (e.g., single data instances, sets of data instances, indexes, etc.) or page components for collecting input (e.g., forms). Links connect pages (and units within pages), and carry the information that enables their computation. Such a modular decomposition into views, areas, pages, and units is particular of WebML, but is a rather classical top-down decomposition.
2. In each site view, the navigation schema of the Website should be expressed as hypertext of named pages, connected by named links, with the obvious interpretation that the user's click on a link causes the loading of the page pointed by the link. Clicking on some links may also trigger back end or external services.
3. For each site views, it should also be possible to express that some pages invoke back-end services [13].

Recently, WebML has been extended for addressing adaptability and context-awareness [17, 19, 20]. The development process for context-aware applications follows the activity flow typically tackled for conventional Web applications. However, some new issues are considered to exploit context, and to achieve adaptive behaviors:

– New modeling abstractions are used to represent mechanisms for context data acquisition, so as the application can rely on a "fresh" representation of the current usage environment.

– New modeling abstractions are adopted to specify the rules for adaptive behaviors based on the acquired context data.

In the rest of this section we will discuss the previous issues, showing the required conceptual modeling extensions.

5.2.1 Identifying areas and pages

The first, coarse-grained step aims to identify the site views to be provided. Starting from the hypertext requirements, a preliminary structure for each site view is then defined, to represent areas and pages which form the hypertext. At this point the designer of context-aware applications also needs to identify those pages and areas that must feature a context-aware behavior and/or must interact with services. As illustrated in Figure 5, our basic assumption is that context-awareness is a property to be associated only to some pages of an application, not necessarily to the application as a whole. Location-aware applications, for example, adapt "core" contents to the position of a user, but "access pages" might not be affected by the context of use. We therefore tag adaptive pages with a "C" label (standing for context-aware), to distinguish them from conventional pages. This label indicates that some adaptability actions must be associated with the page. During application execution these actions must be evaluated prior to the page computation, since they can serve to customize the page content or also to modify the predefined navigation flow. As shown in Figure 5, adaptability actions are specified outside pages, thus they are kept separate with respect to the page specification. The aim is to keep the orthogonality of adaptivity with respect to page and navigation design, and also to highlight the two different logics behind pages and context clouds: while pages act as a provider of contents and services, the context clouds act as modifiers of these contents and services.

In order to continuously evaluate the context state and execute page adaptability actions, the C-pages must be provided with autonomous intervention capabilities. The standard HTTP protocol underlying Web applications implements a strict pull paradigm. Therefore, such capabilities can be achieved by periodically refreshing the viewed page and granting the adaptive logic of the application the possibility to intervene on the application itself

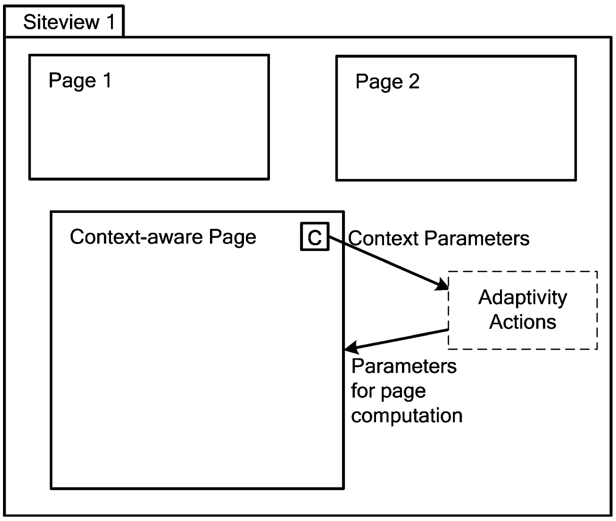

Figure 5 Conventional pages and C-pages in an adaptive hypertext [19].

before rendering the actual response. Polling mechanisms, such as the HTML http-equiv META-option, or also JavaScript, JavaApplets, or Flash scripts, provide a valuable "simulation" of the required active behavior [19]. On one hand they trigger the adaptability actions attached to the page, as required by context model changes; on the other hand, they enable the communication of client-side sensed context data to the application server. When available, such parameters are indeed appended to the page request.

Figure 6 shows the high-level design for two site views in the running case that address the Mobile Camp Operators and the Field Operators. Both the site views include a login page, where users authenticate themselves and are redirected to one of their associated site views. For sake of simplicity, here we assume that only one site view is associated to each user group. Actually, more than one site view can be defined for each group, for example to address the variability of the adopted devices and/or the performed activities. In the case of multiple site views the choice of the site view is realized after login where the user is redirected to after login will depend on the current state of context (i.e., current device, current activity, etc.).

The **Team Tracking** page in the **Mobile Camp Operator** site view, and the **View Map Page** in the **Field Operator** site view are C-labeled. This means that such pages need to be augmented with reactive behaviors, based on some context-data. However the hypertext design annot fully cover all the requirements, and the integration of external services is needed. In this case the hypertext designers send out the list of uncovered requirement to the service designers, and expect the high-level functional description of the external services back from them. However the hypertext designers already include some place-holders within the hypertext schema, to highlight the fact that some pages will also invoke services. In particular, the **Alert Message** page in the **Employee** site view invokes the **SendMessage** service, whose design is illustrated in more details in Section 6.

5.3 High-level service design

The High-Level Service Design phase of UM-MAIS is inspired by the WSMoD methodology. Inputs of this phase are: (a) the data schema, (b) the set of uncovered requirements provided by the hypertext designer, and (c) the shared knowledge of the application domain. According to the WSMoD vision, the design of services is not limited to functional specification only, but also has to consider the set of qualities that have to be offered, and the context in which the service is provided. To manage the complexity of the design, a shared knowledge, with special attention to the non-functional requirements is needed. Ontologies, representing this shared knowledge in a more formal way, seem to be

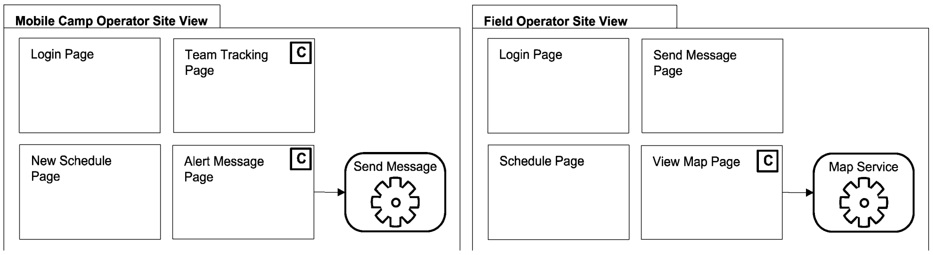

Figure 6 High-level hypertext schema, specifying the organization of two site views in the running example. C-pages and service invocation are sketched.

crucial in providing the rational for making any decision at the different levels of abstraction. To support the service design, three ontologies have been adopted in UM-MAIS:

- Service Ontology (OntoServ in the following)
- Quality Ontology (OntoQoS)
- Context Ontology (OntoContext)

The three ontologies capture all the relevant types of knowledge associated with services, together with semantic relationships among them. In particular *OntoServ* has the purpose of classifying and describing relevant properties of the different services according to the domain characteristics. they include the set of services (both internal and external) that can be used by the service designer. *OntoContext* represents the context model introduced in Section 2, while the most important ontology is *OntoQoS*, that is the result of a unique effort in classifying the qualities that can be associated to services. Currently OntoQoS includes the description of more than 300 domain-independent QoS dimensions [21]. Top-level ontologies are (a) *QoS_Architectural*, including QoS aspects of software and hardware architecture, e.g. screen resolution, (b) *QoS_Network*, representing QoS related to network, e.g. bandwidth, (c) *QoS_Physical*, related to physical aspects of devices, e.g. power consumption, and (d) *QoS_ServiceLevel*, describing QoS intrinsic to Service. QoS_ServiceLevel includes dimensions that are independent from specific domains, such as price, privacy or usability. Moreover, according with specific application domains, OntoQoS can be augmented with domain dependent dimensions.

Starting from the set of uncovered requirements, the service designer identifies abstract services. For each of them, s/he classifies requirements into functional and non functional. In this paper functional means the operations that a service is expected to provide, while the term non-functional (or sometimes the term quality) refers to all the other characteristics associated with a service, related to QoS, user profiles, and delivery channels. The discovery of functional requirements is performed with standard techniques and they will not be described further. The discovery of the non-functional requirements is more difficult; in this case the use of ontologies plays a central role. Browsing the service ontology OntoServ, the service designer can progressively establish the existing service category that the identified services belong to; through service classification, the designer can discover the relations with QoS and the characteristics of services. In a similar way, the designer can exploit OntoContext to identify the context dimensions relevant for services. The designer can decide to adopt the specifications extracted from the ontologies, or to augment them by adding new properties and relations. The list of concepts derived from ontologies is used by the service designer to define a more precise description of both the functional and non functional features of the needed services, that can be used by the hypertext designer in the High-Level Hypertext Design phase. The next step in the High-Level Service Design phase is to evaluate the availability of services that are suitable to provide one or more previously identified operations. OntoServ can help the designer in this analysis, which can lead to three possible results:

– No service exists. In this case the service designer starts the activity of functional modeling of the new services and then sends the new UML class diagrams to the hypertext designer, which can modify navigation according to the functionalities exposed in the UML diagrams.
– There exist services that partially satisfy requirements. The designer extracts the high-level description of the service from available documentation and then, if necessary, integrates that description with the functional specifications the service does not

support, by creating an augmented UML functional class diagram. At the end of this phase the UML class diagram are sent to the hypertext designer.

– Services that fully satisfy requirements exist. The service designer simply sends their functional descriptions (e.g. the WSDL description) to the hypertext designer.

5.3.1 High-level service design in the running case

According to the results of analysis performed in Section 5.2.1 a number of requirements are not fully covered by the hypertext specification such as SR2—Field Operators communicate with the mobile camp, CR1—site information is available to the field operator according to the network conditions available, and CR5—the most reliable and efficient network connection is selected according to the service. The hypertext specification demands the satisfaction of these requirements to services. For the sake of brevity we restrict our discussion only to the above described requirements.

According to SR2 it is possible that a Mobile Camp Operators needs to notify some changes in the working plan to operative teams due unexpected reasons (for example there is the need of special photographic skill in a given site). Operative leader cannot use traditional Web interface, thus there is the need of a flexible notification service able to send the site address (with or without tasks to do) by exploiting different distribution channels. Among others, the use of cellular phones that allow user locating, are considered a strategic objective in order to involve the totality of Field Operators. To support this complex service, a multi-channel flexible notification service, *FlexSend*, has to be developed to deliver messages over different channels, according to specific receiver contexts (preferences, device, and activity).

Figure 7 shows a fragment of OntoServ and OntoQoS, with relationships defined between the two ontologies (dashed lines). According to the WSMoD methodology the service designer has to establish the Web service category that the service belongs to. The Web service is delivered on Internet and can be considered as a special kind of mail service; specifically it is a message delivery service. Consequently, the new service is added in the OntoServ ontology; this means that it inherits all the properties of a generic delivery service. Hence, from the relations with the QoS ontologies, the business expert identifies

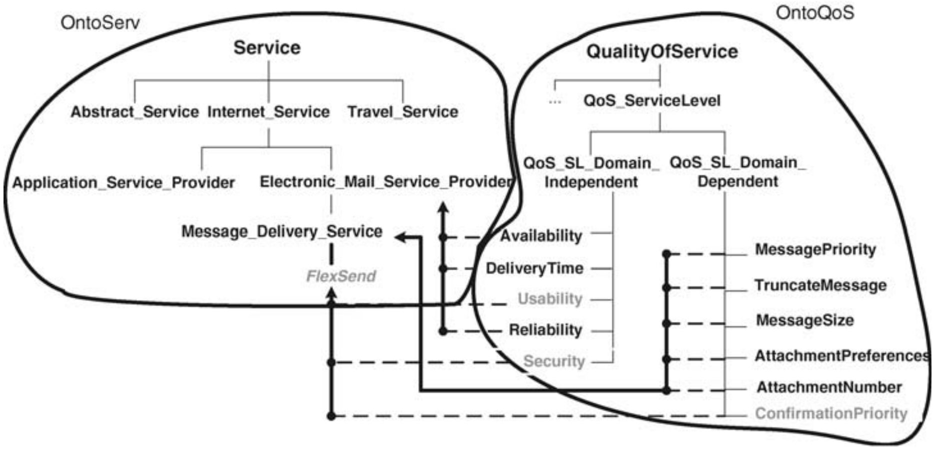

Figure 7 A fragment of OntoServ and OntoQoS and their relationships.

availability and other qualities as relevant for the new service. Moreover, new qualities can be associated with the new service as characterizing quality, such as *usability* (CR1, and CR5).

The service designer now has to associate measures and ranges of values to qualities. According to the type of quality, measures can be expressed by a domain expert as: (a) sets of acceptable values, to be chosen and specified later in the contract between the user and the provider, and (b) constraints, expressed by a minimum or maximum accepted level. In much a similar way, OntoContext supports the identification of meaningful information to define user profiles.

In order to know the exact position of the operative teams, it is necessary to consider mobile devices, to exploit the location capabilities of these devices and related technology. In the following we use the term *Situation* to couple *UserActivity* and *Location*. For example, the field operator according to the day-to-day schedule carries out an activity, (goes to a site, takes photos, etc.) in a specific location, van, sites, mobile camp, etc. For the sake of simplicity, in this paper we consider locations and communication channels as strictly related, even if the relationship is more complex in the real scenario. Hence, a user is requested to select the situation he/she prefers from a list of predefined combinations.

A check among existing services available lets us conclude that no existing services can be exploited in the design of FlexSend. For the *GetAddressBook operation*, we may assume that FlexSend can take advantage of user profiles as described in Section 4 that maintains information about Field Operators.

Figure 8 shows the class diagram that provides an abstract description of the new service. In this phase we are not interested in specifying the channel as a technological component, but only as an abstract resource involved in sending a message. Thus, in the class diagram, derived from the one presented in Figure 8, the designer introduced a *Channel* class characterized by a generic *description* attribute and by the *send()* method. The abstract class *Message* is extended in terms of two classes: *Confirmation Message* and *Requested Message*, which represent the two kinds of messages the Web service is

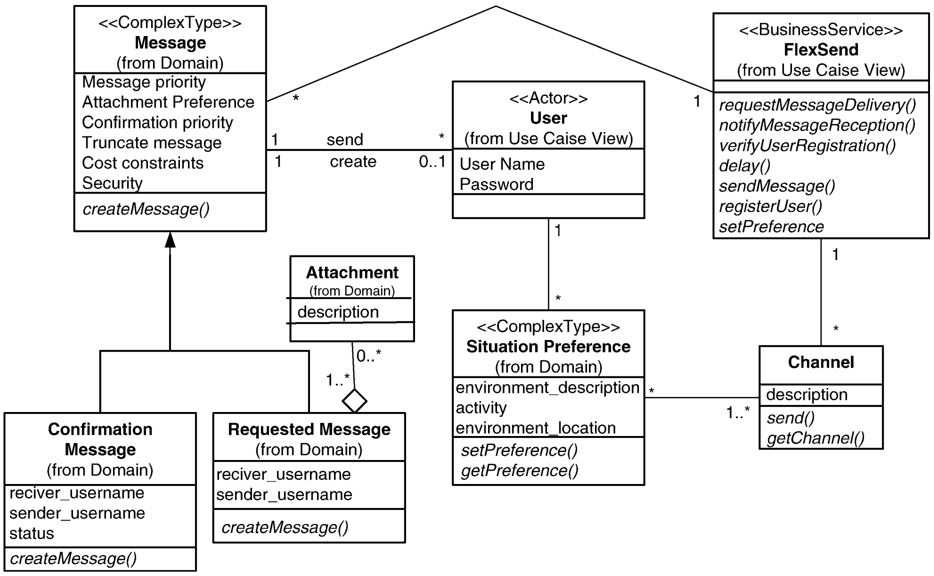

Figure 8 FlexSend class diagram.

requested to deliver. The service designer extends the class Message by adding two specialized classes and by adding basic operation supporting the FlexSend service.

At the end of this phase, the UML class diagram is sent to the hypertext designer who can use it in the definition of the Low-Level Hypertext Design phase. In the class diagram, for example, the hypertext designer will find the input/output parameters needed for *FlexSend* and their types.

6 Low-level hypertext design

The fine-grained design of the hypertext refines the rough schemas produced as output of the coarse-grained design step, by detailing the design of pages and of their possible adaptive behaviors (for C-pages). As outcome of this phase, the application designer produces well formalized hypertext schemas, one for each site view, that serve as reference during the implementation phase.

The design of adaptive behaviors for C-pages assumes a central role. Generally it requires the specification of:

- *Context management operations*, addressing context data acquisition and the consequent context model updating. These are the first actions executed every time a C-page is accessed, to gather an updated picture of the current context.
- *Hypertext adaptability actions*, addressing the definition of actions for page and navigation customization. The event triggering these actions is the page refresh. The most recurrent condition pattern consists in evaluating whether context has changed, hence triggering adaptability actions. Based on the result of condition evaluation, different actions can be triggered:

    - *Adapting page content*. Parameters produced by context data acquisition actions and by condition evaluation can be used for page computation. The result is a page where contents are filtered with respect to the current context.
    - *Adapting navigation*. In some cases, the effect of condition evaluation within the context cloud can be an automatic, i.e., context-triggered, navigation causing the redirection to a different page. The specification of context-triggered navigation only requires the connection of the last link of the context cloud action chain to an arbitrary destination page of the hypertext, to redirect the user to that page. Therefore, links exiting the context cloud and directed to other pages than the cloud's source page represent automatic navigation actions.
    - *Adapting the whole hypertext structure*, to face coarse-grained adaptation requirements, for example due to changes of the user's device, profile and activity within a multi-channel, mobile environment.

For more details about the specification of context management and hypertext adaptability in WebML, the user is referred to [19, 20].

6.1 Low-level hypertext design of the case study

Figure 9 illustrates the detailed design of the **Alert Message** page in the **Mobile Camp Operator** site view, which allows operators to select some field operators belonging to the closest teams, and to send an alert message, asking for an exceptional intervention. In order to adapt the message to the characteristics of the devices currently used by the recipients,

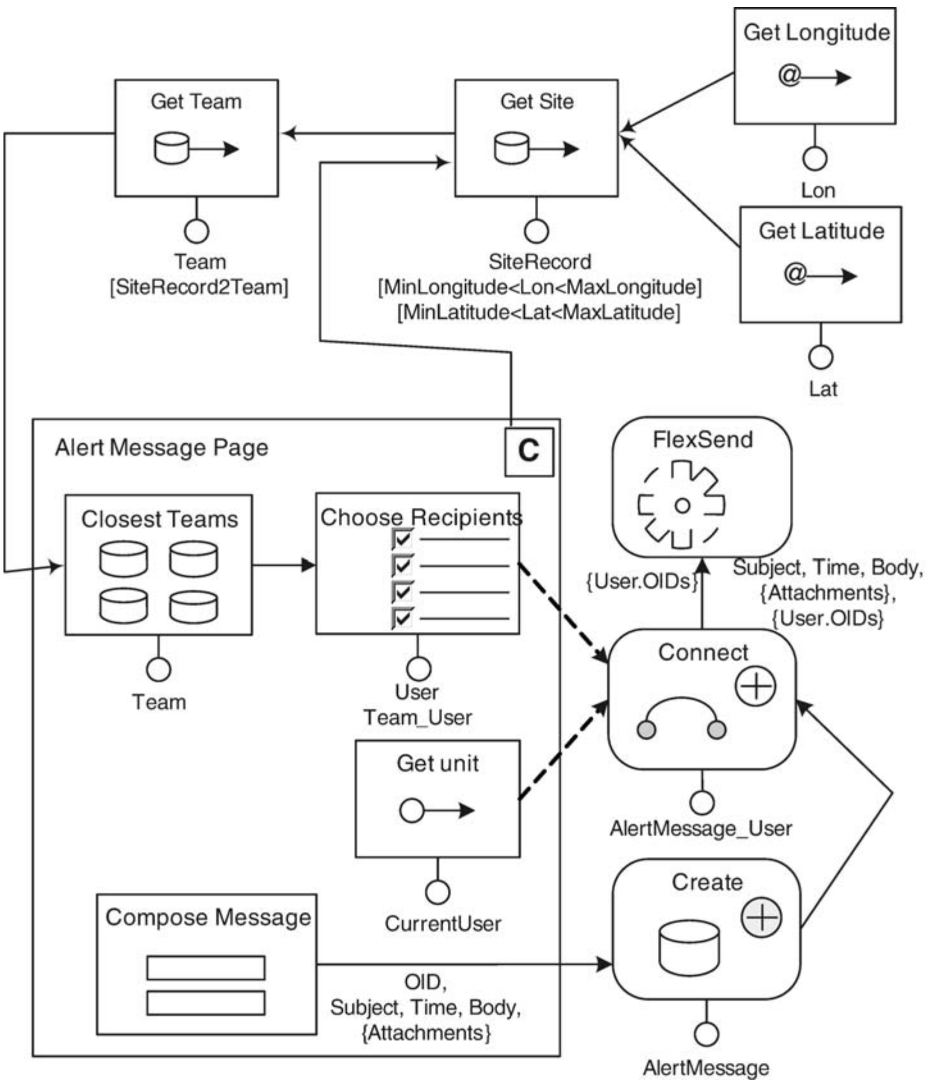

Figure 9 Low-level hypertext design of the Alert Message Composition page.

the message posting is realized through the **FlexSend** service, whose detailed design will be illustrated in Section 7.

The **Closest Teams** unit displays information about the teams that are currently operating close to the mobile camp operator. The selection of such teams is operated by taking into account the current GPS position of the mobile camp operator (**Get Latitude** and **Get Longitude** units), and retrieving the teams that are currently located in the same area (**Get Team** unit) into the data source. The data source maintains an updated representation of the context, also including the current association of teams to their last identified position (see Figure 4). Being the **Alert Message** page a C-page, the operations needed to retrieve the current position of the mobile camp, as well as the team located nearby are executed periodically, according to a specified refresh interval [CDM+06]. This operation constitutes the so-called context cloud for the **Alert Message** page.

The list of the operators working in the closest teams is then displayed (**Choose Recipients** unit), so that the mobile camp operator can choose the users to be alerted. A form (**Compose Message**) allows the camp operator to enter the message subject and body, and possible attachments. When the mobile camp operator pushes the form submit button, an operation chain is activated that uses the data entered in the form for:

- The creation of a new message instance (**Create unit**).
- The association of (through the **Connect unit**) the message to the sender and to the recipients. The sender OID is retrieved by the **CurrentUser** unit, while the recipients OIDs are provided by the **Choose Recipients** index, previously used by the sender for the selection of the Field Operators to be notified. In this way, the application keeps trace of the message, as well as of its sender and recipients.
- Invoking the external **FlexSend** service, whose input is the message previously created and stored in the application data source, its sender and its recipients.

7 Low-level service design

The goal of this phase is to enhance functional UML diagrams generated in the High-Level Service Design with adaptability features related to the QoS, context, and user. The phase is composed of three steps: Adaptability Design, Customization, and Web Service Description.

7.1 Adaptability design

Non functional requirements and the functional UML diagrams defined in High-Level Service Design are inputs to the Adaptability Design phase, whose goal is to include non-functional requirements, user requirements and specification of logical channels into the diagrams. In this phase the designer makes use of the three ontologies, OntoQoS, OntoServ and OntoContext describing quality properties, services, and context. The phase consists of two activities: Data and Operation Design and Interaction Design.

In the **Data and Operation Design** activity, the designer enriches UML class-diagram of services with non-functional requirements by means of the QoSContext UML profile [21]. The designer browses the common data schema, OntoQoS and OntoContext to retrieve both QoS dimensions, user profiles and logical description of distribution channels that will be supported by the service. Channel specifications specialize the channel component defined in the High-Level Service Design phase, by including: (a) the channel components (device, network interface, network, protocols) and their properties, and (b) the range of values and value constraints of logical components. The designer can now evaluate the service, since it has been designed in the previous phase, to verify if it can be provided over a channel. In case of conflicts, the designer redesigns the UML diagrams to fulfill the new quality requirements imposed by channel characteristics and in this case, notifies the new UML functional diagrams to hypertext designer.

A crucial aspect of this activity is the way the system can be aware of the current context for a given user. UM-MAIS adopted the reflective architecture developed in MAIS [4] to supply the designers (and the developers) with reflective components that automatically detect the current context. In this phase, the designer includes the reflective classes that represent the reflective components. Note that the MAIS classes exploited in this paper are independent from any specific implementation, thus preserving the Model Driven approach. Developers can make the choice of adopting the MAIS reflective architecture [2] or replacing it with another reflective mechanism.

The **Interaction Design** activity deals with the enrichment of interaction diagrams (e.g. UML sequence diagrams) with user, channel and QoS requirements previously extracted from ontologies. During this phase the designer evaluates the functionalities of the new service to verify if they satisfy the interaction scenarios and when necessary executes the Data and Operation Design activity to add or change functionalities again. For example, the designer may discover that in order to address the mobile channel, due to a different screen size, information has to be presented with a different format than in PCs. Therefore, since the logical behavior changes, different operations are needed and a new interface has to be conceived to support the new channel.

7.2 Customization

Customization further refines the current design specifications by evaluating the concrete deployment scenario. Design decisions taken so far are compared with the different characteristics of end users and providers to verify the adequacy of service qualities, and user and channel profiles. This process ensures that the delivered service will be effective from both technological and business perspectives. Note that we are still dealing with high-level specifications. The result of the Customization activity is an extended instance of the Platform Independent Model, which includes, functional, non-functional, and contextual aspects, but it is still independent from specific technological details. The Customization activity includes Channel Customization, and User Customization that validate the service specification with respect to actual channel and user profiles.

Channel Customization evaluates the UML diagrams defined so far, with respect to the available technologies. The different concepts represented in ontologies and involved in service design are characterized by a wide set of influence relations. For example, the usability QoS dimension is influenced by comprehensibility, learnability, operability, and pleasantness. In this activity, the influence relations embedded in ontologies are exploited to quantitatively evaluate the quality dimensions with respect to the context for the new service. Such relations can be characterized by composition laws that express how a quality value can be derived from given values of influencing qualities. In the case of linear dependency among QoSs, the Simple Additive Weighting (SAW) technique [33, 58] can be adopted. SAW basically consists in associating a weight with every influencing dimension to express the level of influence in a composition. The score is obtained by summing the weighted values of the influencing dimensions. Weights are normalized, i.e. their sums equal to 1, hence the value of every node in this case is normalized in ranges in [0...1]. However, in the case there is a non linear dependency among QoS, the SAW method cannot be used and thus, a proper composition law (i.e., a function or a table of values) has to be added by domain experts.

To accomplish the task of evaluating the dimensions involved in the design, the activity starts with the extraction from OntoQoS of quality dependency graphs to represent the influence relations between qualities. Nodes of the graph represent ontology instances. Edges among nodes represent influence relations. A dependency graph is often a tree, or it can be transformed into a tree with a top-level quality as root and influencing qualities as children. The transformation consists in duplicating nodes with more then one father. This is justified by the different kind of influence that a quality can have in different cases. On a dependency tree, we apply the following procedure to evaluate a certain Quality Dimension QD:

- Domain experts associate values with leaf nodes of the sub-tree with QD as root.
- Designers compute the value of the root dimension QD, applying the composition laws iteratively.
- Domain experts compare and evaluate results with the current specification assumptions.

For example, the output of this activity could be that the provisioning of a Web service over a GPRS network is inefficient due to slow transmission speed, or that accessing a service via cellular phone cannot meet the desired usability requirements due to screen size and pointing tools.

When the quality analysis leads to unsatisfactory results, the designer has three alternatives: (a) constraints can be relaxed; this computation task is a typical operation research problem that can be automated by means of Integer Programming tools, such as Lp-Solve [32], (b) another distribution channel to provide the service may be selected; (c) the service has to change. When the evaluation does not satisfy business requirements, QoS dimensions directly controlled by the designer are considered. In fact, a QoS dimension can be further classified as dependent on the delivery channel, or dependent on the service-supplier profile. In particular it is:

- Delivery-based QoS, representing the instances of QoS dimensions offered by service provider (e.g. network speed). These QoS dimensions are not directly controllable by the designer and they often represent constraints that he/she has to consider in the design of the Web service.
- Service-based QoS, representing the instances of QoS dimensions that are not offered by the service provider. The designer is responsible for setting specific values for such instances.

The aim of the **User Customization** activity is to provide quality thresholds that are determined by the end user profile stored in the OntoContext ontology. Profiles define service requirements for individual and group users. In order to check the usability of the provided services, specific peculiarities of every user should be highlighted (see [28]).

The quality evaluation of an end user profile starts with the identification of the influence relations between the QoS computed in the Channel Customization and the user profile characteristics derived from OntoContext ontology. Then a User Profile—QoS matrix is defined. The values of the matrix are weights, that is numeric values that represent the level of influence between User Profile and QoS dimensions. Such numeric values are domain dependent and therefore assigned by domain experts. Values are normalized so that the sum of weights of each column is equal to 1.

7.3 Web service description

At this stage of the design process, we need to transform diagrams into Web service descriptions using standard languages. The Web service description step delivers the WSDL and WSOL documents. Final UML diagrams and description documents are the input for the Implementation and Deploy phase that deals with the actual implementation technologies (e.g., .NET, Java) and application server (e.g. JBoss, WebSphere, etc). This step is out of the scope of our methodology. According to the most accepted implementations of the SOA, each Web service has to be described by a WSDL document. WSDL, however, lacks the description of certain aspects such as the QoS offered and the behavior of the Web service [45, 58]. Unfortunately, there are no widely accepted standards to describe such relevant information. Consequently, UM-MAIS directly supports the production of the WSDL and WSOL [51] descriptions but the designer is free to translate the results of previous phases in another Web service description language. The choice of UML as modeling language offers the possibility of describing diagrams in XMI (XML Metadata Interchange). The designer, by means of XSLT technology can translate a UML diagram described in XMI into another XML specification.

OMG provides a guidelines in order to design a UML class diagram able to be translated into WSDL description [43]. The class diagram modeled with the UML profile above defined can be automatically translated into WSDL as shown in the running case. In a similar way we developed a XSL stylesheet able to translate UML diagrams into WSOL specification as shown below.

7.4 Low-level service design in the case study

7.4.1 Adaptability design

Once the service designer has defined the UML class diagrams (Figure 8), in the Data and Operation Design activity, he first identifies functional-aware and level agreement qualities, among the qualities identified in the High-Level Service Design. Functional-aware qualities, e.g. Message priority, Attachment preferences, and so on, have to be evaluated on the diagrams of Figure 8. Moreover personal experience and knowledge supported by ontologies, allow the designer to complete the requirement list by adding new items. In FlexSend the designer introduces *QoS_MessageConstraint* and *QoS_ServiceConstraint* classes to model qualities related to the Message and FlexSend classes. Then these classes can be further refined according to OntoQoS relations. For example, *Usability* that has been selected as a requirement by domain experts can be refined according to the relations stored in ontologies.

A second activity concerns level-agreement qualities, for example privacy in message delivery. Such features are relevant in the case of the inspected archaeological site if it is not aware of illegal excavations. Privacy can be implemented by means of encrypt and decrypt messages and attachments. Moreover, different cryptographic algorithms can implement the privacy level. These requirements lead the modification of class and sequence diagrams to include a new class defining encrypt/decrypt features. The class diagram is also augmented with methods and attributes to complete the specification. The updated version of the class diagram for FlexSend is augmented with functional-aware qualities that have been added as attributes of the message class. A similar approach is used to model instances of the OntoServ, and OntoContext ontologies. The *R_User_Information* class includes meaningful information supplied by the MAIS Reflective Architecture (e.g. available channels and location). Thus information is automatically provided by devices (e.g. mobile phone) equipped with reflective middleware. Figure 10 reports the refined class diagram of FlexSend; for the sake of readability, we report only a one-level refinement for usability.

The actual configuration for the selected channels drives and constraints the design of the new service. In the case study, samples of chosen configurations are:

- C1: Notebook, any interface, Wi-Fi Network, SMTP protocol;
- C2: Personal Computer\Notebook, any interface, any network, SMTP protocol;
- C3: Mobile phone, GSM\UMTS interface, GSM\UMTS, SMS\MMS protocol.

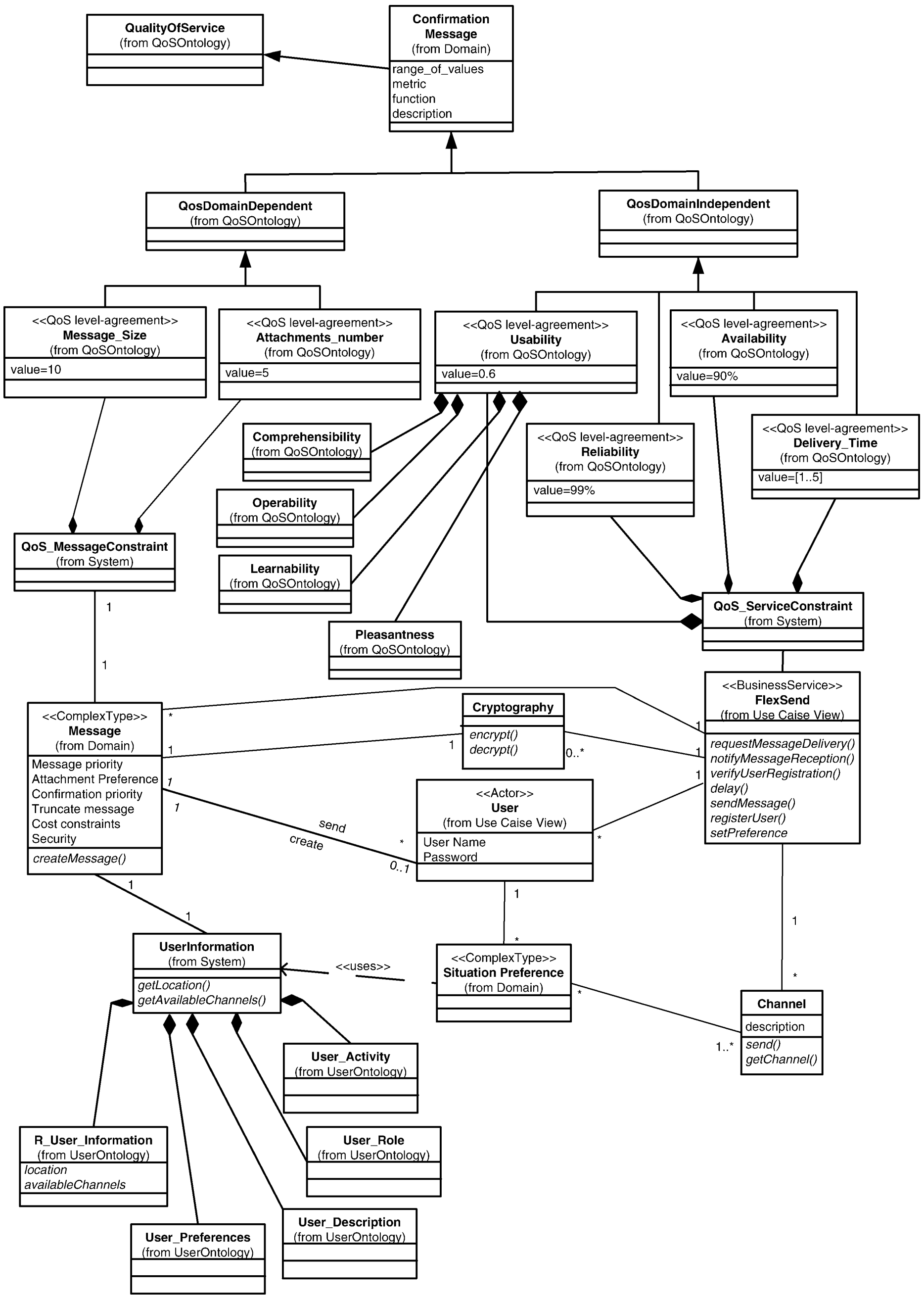
QualityOfService (from QoSOntology)
Confirmation Message (from Domain)
range_of_values
metric
function
description
QosDomainDependent (from QoSOntology)
QosDomainIndependent (from QoSOntology)
<<QoS level-agreement>> Message_Size (from QoSOntology)
value=10
<<QoS level-agreement>> Attachments_number (from QoSOntology)
value=5
<<QoS level-agreement>> Usability (from QoSOntology)
value=0.6
<<QoS level-agreement>> Availability (from QoSOntology)
value=90%
Comprehensibility (from QoSOntology)
Operability (from QoSOntology)
Learnability (from QoSOntology)
Pleasantness (from QoSOntology)
<<QoS level-agreement>> Reliability (from QoSOntology)
value=99%
<<QoS level-agreement>> Delivery_Time (from QoSOntology)
value=[1..5]
QoS_MessageConstraint (from System)
QoS_ServiceConstraint (from System)
<<BusinessService>> FlexSend (from Use Caise View)
requestMessageDelivery()
notifyMessageReception()
verifyUserRegistration()
delay()
sendMessage()
registerUser()
setPreference
Cryptography
encrypt()
decrypt()
<<ComplexType>> Message (from Domain)
Message priority
Attachment Preference
Confirmation priority
Truncate message
Cost constraints
Security
createMessage()
<<Actor>> User (from Use Caise View)
User Name
Password
send
create
UserInformation (from System)
getLocation()
getAvailableChannels()
<<uses>>
<<ComplexType>> Situation Preference (from Domain)
Channel
description
send()
getChannel()
User_Activity (from UserOntology)
User_Role (from UserOntology)
R_User_Information (from UserOntology)
location
availableChannels
User_Preferences (from UserOntology)
User_Description (from UserOntology)
1
*
0..*
0..1
1..*

Figure 10 A fragment of class diagram describing QoS refinements.

Table 2 Example of a channel configuration.

Characteristic	Value
Resolution	176×208
Size	2.0″
BitsForPixel	24
ColorCapable	Yes
ImageCapable	Yes

Due to the fact the exposed interface of FlexSend is not modified, there is no need for further interactions with the hypertext designer. In the Interaction Design activity the designer defines and models the interaction between users and service. In FlexSend, the sender issues a request to send a message. The service behavior depends on sender parameters and on receiver context. For example, if there are no available channels to deliver the message according to the requested QoS dimension, the service applies one of the predefined delivery policies.

7.4.2 Customization

Following the running case we show how to check if the value of usability obtained is compliant with the business requirements. As we said, u*sability* depends on *Comprehensibility, Learnability, Operability and Pleasantness*. Furthermore, *Comprehensibility* depends on *ScreenQoS* and *SoundQoS*. Finally, *ScreenQoS* depends on the quality attributes of the screen device (resolution, size, etc).

To evaluate a quality, on one hand we need to consider the requirements stated in the High-Level Service Design phase, on the other hand a possible configuration for actual channels has to be selected. In our case, business experts have requested a *Medium* level of usability. Hence, a *usability* value '*medium at least*' can be regarded as *usability*>0.6. At this point, the selection of one or more channel configurations is performed. For example, in OntoContext the device component class is composed of hardware and software components, the hardware components

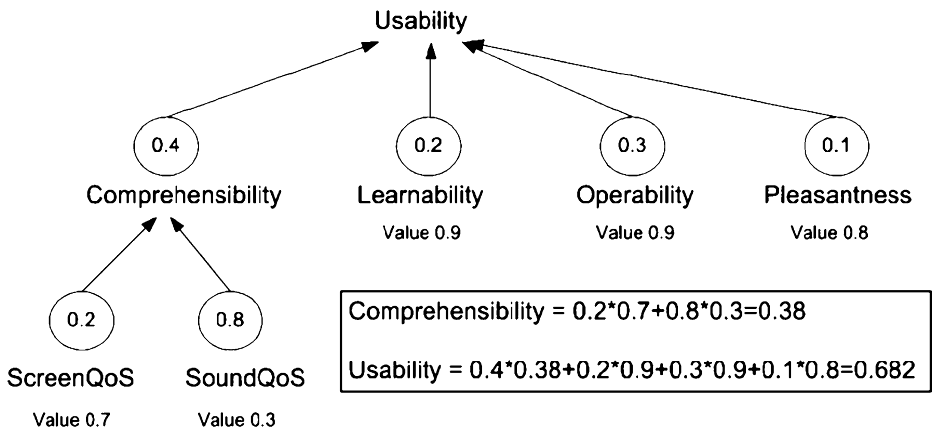

Figure 11 SAW method.

Table 3 User Profile-QoS values matrix.

User profile/QoS	Comprehensibility	Learnability	Operability	Pleasantness
Skills		0.2	0.3	
Body function	0.6	0.4	0.3	
Expertise		0.3		
Education	0.4	0.1	0.1	
Relation capabilities			0.3	1

are CPU, input system, display, sound system, memory, and power. Moreover each sub-component is described in terms of its technological characteristics.

We need to assign values to define the technical characteristics of the device display. An example of configuration is given in Table 2. As we did for usability requirements, we assign a normalized value to ScreenQoS *equal to* 0.7.

Figure 11 shows how to evaluate *Usability* and *Comprehensibility* when the device selected for the end user provides ScreenQoS=0.7 and SoundQoS=0.3. In this case, we obtain that *Comprehensibility* is equal to 0.38 and *Usability* is equal to 0.682. This value satisfies our design hypothesis and so any further activity is not necessary.

At this point, the influence relations are used to define a User profile/QoS matrix. Table 3 reports the User Profile/QoS matrix for the FlexSend case study. We assume that the value of *Body function* and *Education* are equal to 0.6 and 0.4 so the value of *Comprehensibility*, obtained by summing weighted values is 0.76. The value represents the threshold of usability requested by users of the FlexSend service. The actual version of the Web service generates a comprehensibility value, as shown in Table 3, that does not achieve the threshold and so the design assumptions have to be revised.

Analyzing the QoS tree, *Comprehensibility* is related only to channel device QoS (*ScreenQoS* and *SoundQoS*). Qualities related to the actual version of FlexSend service do not influence this QoS. So, *Comprehensibility* is classified as delivery-based QoS. Thus the designer must return to the previous step, identify the most violated constraints and select new technical configuration. If this change does not satisfy the business constraints, the designer has to go back to the previous step and change UML diagrams describing the Web service.

7.4.3 Web service description

In the following, we show an example of WSDL sections that are generated using the UMT tool and the class diagram produced in the Customization step. The Message class is translated as a ComplexType in the <type> section of the WSDL document (Figure 12). The

```
<types>
  <xsd:complexType name="Message">
    <xsd:sequence>
       <xsd:element name="Message Priority" type="xsd1:String">
       <xsd:element name="Attachment_preference" type="xsd1:String">
       <xsd:element name="Confirmation priority" type="xsd1:String">
       <xsd:element name="Truncate message" type="xsd1:String">
       <xsd:element name="Cost constraints" type="xsd:int">
       <xsd:element name="Security"  type="xsd1:String">
     </xsd:sequence>
   </xsd:complexType>
 </types>
```

Figure 12 ComplexType of FlexSend.

Table 4 Adopted tools in UM-MAIS.

Tools/Phase	Office automation	WebML run time	Protege	UML modeller	LP-Solve	UMT
Requirements management	x					
Data design		x		x		
High level hypertext design		x				
High level service design			x	x		
Low level hypertext design		x				
Low level service design				x	x	x
Deployment and development		x				

ComplexType has the same name of the class and the composed part elements represent all the class attributes. Every attribute is shown with the same name and the same type of UML attributes (e.g String, Int, Boolean, etc.). The public methods of the class stereotyped as ≪BusinessService≫ are translated as both a WSDL portType and a WSDL binding.

8 Supporting tools

The effectiveness of a methodology is strictly related to the availability of support tools, models, and artifacts that can be easily used in third party solutions. In order to enhance interoperability and practical applications, UM-MAIS is supported by an integrated suite of widely used tools. In this way it is also possible to reduce the well-known learning-curve problem and resistance-to-change attitude. Table 4 shows the adopted tools for each phase. The requirement analysis is supported by typical office automation tools that can be easily adopted by analysts.

Concerning the Data design phase E/R or UML modelers help designers in the definition of the common data schema. Both High and Low-Level Hypertext Design have been embedded into a complete development framework for context-aware adaptive Web information systems, by means of an adaptive deployment platform. It allows also adapting presentation properties of the generated application [42]. As shown in Figure 13, starting from extended WebML schemas, developed with the WebML run time environment [55], XML-formatted "pages" (annotated XML documents) containing presentation-related properties are generated and refined at each page request.

At runtime the properties are evaluated by a dedicated Rule Engine (rules are formalized as XSLT transformations) that takes in input properly formatted XML documents and

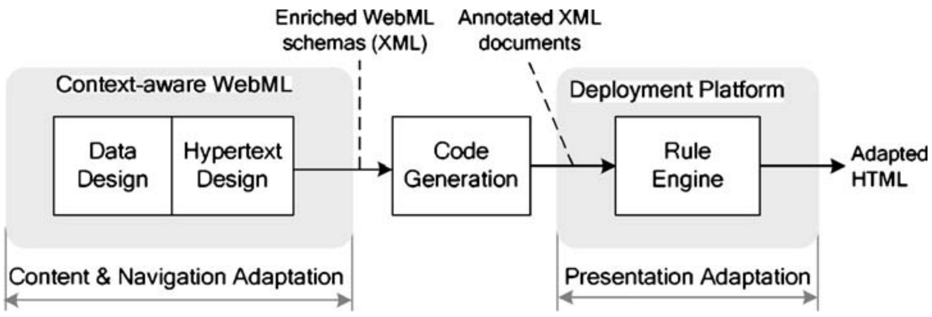

Figure 13 UM-MAIS approach to modeling and deploying context-aware front-ends.

produces in output HTML pages adapted to the current context conditions. This seamless integration of conceptual modeling techniques and technological results allows the specification of both content and navigation adaptations on one hand, and style or layout adaptations on the other hand at their right level of abstraction.

Both High- and Low-Level Service Design use Protégé, a popular free-software tool originally developed at Stanford University for knowledge acquisition [40, 41]. Protégé allows for the editing of ontologies and building of knowledge bases with a user-friendly interface and interactive environment. Protégé supports OWL, RDF schema, XML files with a DTD, and XML schema files. Service designers can navigate the ontologies and/or augment them to fulfill their requirements. High-Level Service Design phase, Adaptability and Customization steps of Low-Level Service Design adopt well-known design tools such as Rational Rose as the UML modeler tool and Protégé tool to exploit ontologies. In the Customization step we adopt Lp-solve [32] as Integer Linear Programming (see the Section 7.2). In the Web Service Descriptions step, we have exploited UMT [52], a tool that provides for model transformation and code generation based on UML models in the form of XMI. XMI models are imported by the tool and converted to a simpler intermediate format, which is the basis for validation and generation towards different target platforms. The intermediate format is an XML format, which in UMT is called XMI Light. UMT transforms the XMI Light to the desired target technology using a transformer. In UMT, this is normally XSLT, but it may also be a Java-implementation. We have extended this tool by adding an XSLT stylesheet in order to perform the transformation of UML diagrams into WSOL specifications. To support the automatic generation of Web service descriptions and to provide a unique way to make use of ontologies in the design of UML diagrams we defined a specific UML profile to identify the core service elements, QoS dimensions and channel specifications.

The development and deployment phase is supported by the WebML run time tool and by typical Web service development environments such as IBM Websphere.

9 Conclusion

This paper has shown a methodological framework supporting the design of Multichannel Adaptive WISs. We have argued that no comprehensive approach covers all the needs of this new end relevant class of Web applications; we have then motivated the main choices underlying UM-MAIS, a methodology that fills this gap. UM-MAIS enriches three existing methodologies devoted for requirement analysis (AWARE), Multichannel Web interface (WebML), and Multichannel services (WSMoD). We have specifically focused on the methodological steps that cope with the support of heterogeneous devices and channels and the adaptation to the context (including individual user profiles). These steps have been illustrated in detail and shown at work on a running example.

We plan to extend further the methodology in order to cover the deployment and testing phase, with special attention to the definition of architectural solutions and their impact on the design choices about Web services. We also plan to provide a unique software framework that supports all methodological phases, by integrating the tools that are already available in the context of the AWARE, WebML, and WSMoD methodologies.

Acknowledgement This work has been partially supported by the Italian MURST-FIRB Project MAIS (Multi-channel Adaptive Information Systems) and by the MURST-FIRB Project eG4M (electronic Government for Mediterranean Countries).

References

1. Abrahamsson, P., Warsta, J., Siponen, M.T., Ronkainen, J.: New directions on agile methods: a comparative analysis. In: Proceedings of the 25th International Conference on Software Engineering (Portland, Oregon, May 03–10, 2003). International Conference on Software Engineering, pp. 244–254. IEEE Computer Society, Washington, DC (2003)
2. Adorni, M., Arcelli, F., Ardagna, D., Baresi, L., Batini, C., Cappiello, C., Comerio, M., Comuzzi, M., De Paoli, F., Francalanci, C., Grega, S., Losi, P., Maurino, A., Modafferi, S., Pernici B., Raibulet C., Tisato, F.: The MAIS approach to Web service design. In: Proceedings of Tenth International Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD) (2005)
3. Antón, A.I.: Goal-based requirements analysis. In: Proceedings of the Second International Conference on Requirement Engineering RE '96 (1996)
4. Arcelli, F., Raibulet, C., Tisato, F., Adorni, M.: Architectural reflection in adaptive systems. In: SEKE 2004, pp. 74–79 (2004)
5. Barna, P., Houben, G.-J., Frasincar, F.: Specification of adaptive behavior using a general-purpose design methodology for dynamic web applications. In: AH'04—Proceedings of Adaptive Hypermedia, pp. 283–286 (2004)
6. Baumeister, H., Knapp, A., Koch, N., Zhang, G.: Modelling adaptivity with aspects. In: ICWE 2005, pp. 406–416
7. Benatallah, B., Sheng, Q.Z., Dumas, M.: The Self-serv environment for web services composition. IEEE Internet Computing (2003)
8. Binemann-Zdanowicz, A., Kaschek1, R., Schewe1, K., Thalheim, B.: Context-aware web information systems. In: Proceedings of the First Asian-Pacific Conference on Conceptual Modeling, Dunedin, New Zealand, pp. 37–48 (2004)
9. Boehm, B., Port, D., Abi-Antoun, M., Egyed, A.: Guidelines for the Life Cycle Objectives (LCO) and the Life Cycle Architecture (LCA) Deliverables for Model-based Architecting and Software Engineering (MBASE), USC technical report USC-CSE-98-519, Los Angeles, CA, 90089, 1999
10. Boerstler, J., Janning, T.: Traceability between requirements and design: a transformational approach. In: Proceedings of 16th International Computer Software & Application Conference, pp. 362–368 (1992)
11. Bolchini, D., Mylopoulos, J.: From task-oriented to goal-oriented web requirements analysis. In: Proceedings of International Conference on Web Information System Engineering WISE'03, Rome, Italy (2003)
12. Bolchini, D., Paolini, P.: Goal-driven requirements analysis for hypermedia-intensive web applications, In: Requirements Engineering Journal, RE'03 special issue, Springer, Berlin (2004)
13. Brambilla, M., Ceri, S., Comai, S., Fraternali, P., Manolescu, I.: Model-driven specification of web services composition and integration with data-intensive web applications. IEEE Data Eng. Bull. **25**, 53–59 (2002)
14. Cappiello, C., Comuzzi, M., Mussi, E., Pernici, B.: Context management for adaptive information systems. Electr. Notes Theor. Comput. Sci. **146**(1), 69–84 (2006)
15. Central institute for conservation (Istituto centrale per il restauro), the Risk map of cultural heritage, verified 28 September 2006, http://www.icr.arti.beniculturali.it/rischio/rischio00e.htm
16. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-intensive Web Applications. Morgan Kauffmann (2002)
17. Ceri, S., Daniel, F., Matera, M.: Extending WebML for modeling multi-channel context-aware web applications. In: Proceedings of WISE'03 Workshops, pp. 225–233, IEEE Press, Rome, Italy, (December 2003)
18. Ceri, S., Fraternali, P., Bongio, A., Butti, S., Acerbis, R., Tagliasacchi, M., Toffetti, G., Conserva, C., Elli, R., Ciapessoni, F., Greppi, C.: Architectural issues and solutions in the development of data-intensive web applications. In: Proceedings of CIDR 2003, Asilomar, CA, USA (January 2003)
19. Ceri, S., Daniel, F., Facca, F., Matera, M.: Model-driven engineering of active context-awareness. In: Atzeni, P., Catarci, T., Pernici, B. (eds.) Special issue on multi-channel adaptive information systems, WWW Journal (2007)
20. Ceri, S., Daniel, F., Matera, M., Facca, F.: Model-driven development of context-aware web applications. ACM Transactions on Internet Technology **7**(1), (2007), January
21. Comerio, M., De Paoli, F., Grega, S., Maurino, A., Batini, C.: WSMoD: a methodology for QoS-based web service design. International Journal of Web Service Research **2**, (2007)
22. Dardenne, A., Fickas, S., van Lamsweerde, A.: Goal-directed concept acquisition in requirements elicitation. In: Proceedings IWSSD'91, Como (1991)
23. De Bra, P.: Adaptive educational hypermedia on the web. Commun. ACM **45**(5), 60–61 (2002), May
24. Fiala, Z., Hinz, M., Houben, G.-J., Frasincar, F.: Design and implementation of component-based adaptive web presentations. In: ACM SAC'04, pp. 1698–1704 (2004)

25. Fraternali, P.: Tools and approaches for developing data-intensive web applications: a survey. ACM Comput. Surv. **31**(3), 227–263 (1999)
26. Gómez, J., Cachero, C., Pastor, O.: Conceptual modeling of device-independent web applications. IEEE multimed. **8**(2), 26–39 (2001)
27. Gotel, O., Finkelstein, A.: An analysis of the requirements traceability problem. In: Proceedings of International Conference on Requirements Engineering (ICRE), pp. 94–101, IEEE CS Press (1994)
28. Graziani, P., Billi, R., Burzagli, L., Gabbanini, A., Palchetti, B., Bertini, E., Kimani, S., Sbattella, L., Barbieri, T., Bianchi, C., Batini, C.: Definition of user typologies. MAIS internal report R7.3.1 (2003)
29. Groenmo, R., Skogan, D., Solheim, I., Oldevik, J.: Model-driven web services development. International Journal of Web Services (2004)
30. Kappel, G., Proll, B., Retschitzegger, W., Schwinger, W.: Customization for ubiquitous web applications a comparison of approaches. International Journal of Web Engineering and Technology (2003)
31. Koch, N., Kraus, A., Hennicker, R.: The authoring process of the UML-based web engineering approach. In: First International Workshop on Web-oriented Software Technology (IWWOST01) (2001)
32. LP_SOLVE.: Linear Programming Code, March 1 2006. Available at: http://www.cs.sunysb.edu/~algorith/implement/lpsolve/implement.shtml (2005)
33. Lum, W.Y., Lau, F.C.M.: User-centric content negotiation for effective adaptation service in mobile computing. IEEE Trans. Softw. Eng. 1000–1111 (2003)
34. Manolescu, Ioana, Brambilla, Marco, Ceri, Stefano, Comai, Sara, Fraternali, Piero: Model-driven design and deployment of service-enabled web applications. ACM Transactions on Internet Technology **5**(3), 439–479 (2005)
35. Maurino, A.: Using mobile information systems for automatic data acquisition of complex archaeological sites. International Conference on Computer Applications and Quantitavie Methods in Archaeology, Tomar (2005)
36. Maurino, A., Modafferi, S.: Challenges in designing of cooperative mobile information systems for the risk map of Italian cultural heritage. In: First Workshop on Multichannel and Mobile Information Systems held in conjunction with WISE, Rome (2003)
37. Mylopoulos, J., Lau, D.: Designing web services with tropos. In: Proceedings of International Conference on Web Services (2004)
38. Mylopoulos, J., Chung, L., Yu, E.: From object-oriented to goal-oriented requirements analysis. Commun. ACM **42**(1), 31–37 (1999)
39. Nawrocki, J.R., Jasiñski, M., Walter, B., Wojciechowski, A.: Extreme programming modified: embrace requirements engineering practices. In: International Conference on Requirement Engineering, Germany, pp. 303–310 (2002)
40. Noy, N.F., Fergerson, R.W., Musen, M.A.: The knowledge model of Protégé—2000: combining interoperability and flexibility. In: EKAW, pp. 17–32 (2000)
41. Noy, N.F., Sintek, M., Decker, S., Crub´ezy, M., Fergerson, R.W., Musen, M.A.: Creating semantic web contents with Protégé—2000. IEEE Intell. Syst. **16**, 60–71 (2001)
42. Pernici, B. (ed.): Mobile Information Systems, Infrastructure and Design for Adaptivity and Flexibility. Springer, Berlin (2006)
43. Radhakrishnan, R., Wookey, M.: Model driven architecture enabling service oriented architectures. 1 March 2006, http://www.omg.org/news/whitepapers/mdasoa.pdf (2004)
44. Schwabe, D., Guimaraes, R., Rossi, G.: Cohesive design of personalized web applications. IEEE Internet Computing **6**(2), 34–43 (2002)
45. Shuping, R.: A framework for discovering web services with desired quality of services attributes. In: Proceedings of International Conference of Web Services (ICWS), pp. 208–213 (2003)
46. Siegel, J.: OMG Staff Strategy Group. Developing in OMG's Model-driven Architecture (2001)
47. Skogan, D., Gronmo, R., Solheim, I.: Web service composition in UML. In: Proceedings of the Eighth IEEE Intl Enterprise Distributed Object Computing (EDOC) (2004)
48. Stephenson, J.: Service Oriented Architecture, OptimalJ, CBDI Report, Retrieved March 1 2006 http://www.omg.org/mda/mda_files/CBDI-SOAOptimalJ-US2003.pdf (2003)
49. Thalheim, B., Dusterhoft, A.: SiteLang: Conceptual modeling of internet sites in conceptual modeling In: ER 2001, Yokohama, Japan (2001), November 27–30
50. Torlone, R., et al.: Methods and tools for the development of adaptive applications. In: [43], pp. 209–247
51. Tosic, V., Patel, K., Pagurek, B.: WSOL Web Service Offerings Language, CAiSE 02/WES '02: Rev. Papers from the International Workshop on Web Services, E-business, and the Semantic Web, pp. 57-67, London, UK (2002)
52. UMT.: SINTEF, UML Model Transformation Tool, Retrieved March 1 2006 http://umt-qvt.sourceforge.net (2005)
53. van Lamsweerde, A.: Goal-oriented requirements engineering: a roundtrip from research to practice. In: Proceedings of the Requirements Engineering Conference, 12th IEEE International (RE'04) (2004)

54. Vdovjak, R., Frasincar, F., Houben, G.-J., Barna, P.: Engineering semantic web information systems in Hera. J. Web Eng. **2**(1–2), 3–26 (2003)
55. WebRatio.: http://www.webratio.com/ (Accessed in September 2006)
56. Yu, E.: Towards modelling and reasoning support for early-phase requirements engineering. In: Proceedings of IEEE International Conference on Requirements Engineering RE'97 (1997)
57. Yu, E., Mylopoulos, J.: Why goal-oriented requirements engineering. In: Proceedings of the Fourth International Workshop on Requirements Engineering: Foundation for Software Quality, Pisa, Italy (1998)
58. Zeng, L., Benatallah, B., Ngu, A., Dumas, M., Kalagnanam, J., Chang, H.: QoS-aware middleware for web service composition. IEEE Trans. Softw. Eng. **30**(5), 311–327 (2004)