Journal of Web Engineering, Vol.2, No.3 (2004) 193-212

REQUIREMENTS ENGINEERING FOR WEB APPLICATIONS – A COMPARATIVE STUDY

M. JOSÉ ESCALONA

University of Seville. Spain

escalona@lsi.us.es

NORA KOCH

University of Munich (LMU) and

F.A.S.T. GmbH, Germany

kochn@informatik.uni-muenchen.de

koch@fast.de

Received August 9, 2003
Revised January 9, 2004

The requirements engineering discipline has become more and more important in the last years. Tasks such as the requirements elicitation, the specification of requirements or the requirements validation are essential to assure the quality of the resulting software. The development of Web systems usually involves more heterogeneous stakeholders than the construction of traditional software. In addition, Web systems have additional requirements for the navigational and multimedia aspects as well as for the usability as no training is possible. Therefore a thoroughly requirements analysis is even more relevant.

In contrast, most of the methodologies that have been proposed for the development of Web applications focus on the design paying less attention to the requirements engineering. This paper is a comparative study of the requirements handling in Web methodologies showing trends in the use of techniques for capturing, specifying and validating Web requirements.

Keywords: requirements engineering, Web methodology, survey
Communicated by: D Schwabe & O Pastor

1 Introduction

The intensive use of Web Applications has produced, among others, a rising interest in the development of methodological approaches providing a suitable support for the construction of Web applications. Several research groups proposed methodologies with processes, models and techniques to build such applications [34, 18, 32, 9] in the last years. However, if we analyze these different approaches, most of them focus on the design workflow in the life cycle, while other tasks like requirements engineering, tests and quality management are handled with less relevance or not included at all.

In the development of traditional (non-Web) applications both practitioners and process experts regard requirements engineering as the most important phase in the development process since the most common and time-consuming errors as well as the most expensive ones to repair, are errors usually consequence of an inadequate engineering of requirements [39]. Many techniques have been proposed. There are specific ones for the capture of requirements, such as interviewing or

storyboarding, techniques for the specification of the requirements, such as scenarios or use case modeling, and for the validation of the elicited requirements, such as prototyping.

Although the relevance of requirements engineering is well known these techniques are poorly applied in the Web engineering field. We stress that on the contrary, Web applications require a more extensive and detailed requirements engineering process due to the number of stakeholders involved and due to the diversity of the requirements including among others requirements on the navigation and on the business processes as well as Web usability. It is always an iterative process.

The study performed by Barry and Lang [2] showed that practitioners find development difficult and that there is an increasing demand for them to deliver high-quality Web-based software products in-budget and in-time. They urge to find solutions for user-centered approaches which translate users' navigational requirements into system representations. Modeling techniques that aid in requirements representation and communication will be essential part of the future CASE Tools used in the development of Web applications.

The motivation for this work is to show the deficiencies that the current Web methodologies present and on the same time offer a palette of requirements engineering techniques which could aid Web developers in their work. In addition, the comparison presented should help in the continuous process of improvement of the existing Web methodologies and their tool support in order to focus more on requirements engineering, and therefore contribute to improve the quality of the Web applications that are built with these methodologies.

The present work gives a survey and a comparative study of the current approaches available in the Web field that use different techniques and model to handle requirements engineering[a]. For that reason, we outline the requirements engineering process and an overview of classic requirements engineering techniques in Section 2. The brief description includes the most commonly used techniques to capture, define and validate the requirements of a system. In Section 3, the main Web methodologies are described including requirements specification, that in different degree of detail include requirements specification. This section includes also a classification of requirements. In Section 4, these approaches are classified and compared from different points of view. Finally, in Section 5 are presented some conclusions and future works.

2 Requirements Engineering Techniques

A requirement is defined as a condition or capability that must be met or fulfilled by a system to satisfy a contract, standard, specification, or other formally imposed documents (IEEE Standard 610.12-1990). The requirements defined for a system should be: correct, consistent, verifiable and traceable. Requirements engineering is the process of eliciting, understanding, specifying and validating customers' and users' requirements. It also identifies the technological restrictions under which the application should be constructed and run. It is an iterative and co-operative process with the objective to analyze the problem, to document the results in a variety of formats and evaluate the precision of the results produced [11].

Whenever a software application is built, be it for the Web or not, the development team has to acquire certain knowledge about the problem domain and the application's requirements. The

[a] This work has been partially granted by the Deutscher Akademischer Austauschdienst (DAAD).

elicitation and specification of these requirements is a complex process as it is necessary to identify the functionality that the system has to fulfill in order to satisfy the users' and customers' needs.

Although there is a lack of a standardized process supporting requirements handling and guaranteeing the quality of the results, best practice in the development of general software applications provide a set of techniques. Such techniques are also recommended by some Web methodologies for requirements specification of Web applications. It is important to note, however, that the selection of appropriate techniques belongs to the responsibility of the development team and the success of the results depends on this team, the group of customers and users that participate in the process.

The iterative process of requirements engineering consists of three main activities [24]:

- ✓ requirements elicitation
- ✓ requirements specification
- ✓ requirements validation

Figure 1 shows this process of requirements engineering. It is represented as a UML activity diagram [35] and is part of the iterative development life cycle, which in the case of Web applications has the tendency to continue during the whole life of the application. Sawyer and Kotonya [33] describe a requirements engineering process that includes a fourth activity: the requirements analysis and negotiation. We consider requirements analysis as part of the requirements specification.

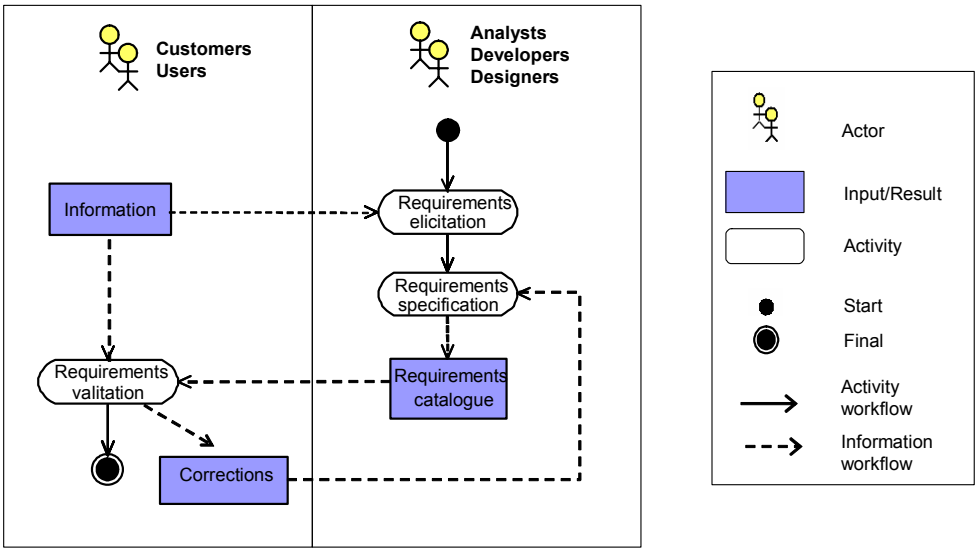

Figure 1: The Requirements Engineering Process

The process starts with the requirements elicitation. The set of developers collect information from the users and customers. Information can be gathered from different sources, such as documents, legacy applications, interviews, etc. which are used in the preparation of the requirements catalogue. Finally, the requirements validation is performed to find out if there are some inconsistencies, mistakes

or undefined requirements. The specification-validation process is iterative and may be executed several times in complex projects.

In the next sections, we briefly describe some classic techniques to elicit, specify and validate requirements. These techniques can be more or less suitable for requirements engineering in the Web environment. It is very difficult to establish precise criteria to select the most suitable techniques. These criteria may include the easiness of learning and using the technique, its scalability, its cost, the quality of its results and the time required for its application. For example, the use of natural languages in the specification of requirements gives less precise results than a description done using use cases, which as well are less precise than requirements described using formal languages. Techniques like JAD are more time-consuming and more difficult to use than other techniques like interviewing, but they produce results of higher quality.

2.1 Requirements Elicitation

The capture or the elicitation of requirements is the activity by means of which the development team collects from any available source the functionality the system needs to provide to the future users. This topic covers what sometimes is termed as requirements capture, requirements discovery or requirements acquisition. The process of requirements elicitation can be complex, mainly if the problem domain is unknown for the analysts. Thus, a set of techniques have been defined and tested by requirements engineering experts to make this step more efficient and precise.

In the remaining of the section we present an overview of the most relevant techniques used in requirements elicitation in the context of a standard software development process.

- **Interviewing** is a traditional and frequently applied technique. By means of interviews analysts are able to understand the problem and get information about the objectives of the application to be developed. The interviewing technique and certain guidelines of how to use them correctly are described in detail by Durán, Bernáldez, Ruíz and Toro [8] as well as Pan, Zhu and Johnson [28]. Basically, the interviewing process covers four steps: the identification of stakeholders for the interview, the preparation of the interview, the interview itself and the documentation of the results in form of an interview protocol.

  Interviews are not easy to perform; they require a vast experience of the interviewer who needs to have the ability to choose the most suitable interviewees [28].

- **JAD (Joint Application Development)** can be regarded as an alternative to interviewing. It is a group technique that requires the participation of all stakeholders of a project, i.e. analysts, designers, users, system administrators and customers [23]. The requirements are captured in a set of sessions over several days. In each session, the high level requirements are analized and the problem field and the documentation are established. During each session the group discusses about the different topics, drawing and documenting, as a result a set of documented conclusions. Such conclusions drive the specification of the system requirements. JAD is based on four basic principles: group dynamic, the use of visualization techniques to improve communication, the support of an organized and rational process, and a philosophy of documentation of type WYSIWYG (What You See Is What You Get). On every JAD session the requirements of the system are becoming more concrete.

This technique provides several advantages compared to interviewing mainly because it saves time. In JAD it is not necessary to compare customer's opinions with one another. Conversely, JAD needs a good integrated and organized group of stakeholders.

- **Brainstorming** is also a group meeting technique similar to JAD. It consists of collecting non-evaluated ideas and information of all stakeholders of the project [30]. The number of participants of such brainstorming meetings should not exceed 10 (stakeholders of the project); one of them has to assume the role of moderator, but should not control the session.

  In contrast to JAD, brainstorming is easier to use as it requires less work in the group. Moreover, as brainstorming often provides a better overview of the system requirements, it is frequently used in first meetings where concrete details are not still needed.

- **Concept Mapping** is a technique by means of which concept maps are built [28]. Concept maps are graphs with vertexes representing concepts and edges representing relationships between these concepts. These graphs, developed by the project team together with the customer and/or final user, are frequently used as a simple communication medium, mainly because they are written in the customer's language. However, some care is required to avoid a subjective and ambiguous description of complex systems. It is recommended to provide an additional textual description.

- **Sketching and Storyboarding** is a technique frequently used by graphical designers in the development of Web applications. It consists of a schematic representation (usually on the paper) of the different user interfaces (sketches). These sketches can be grouped and connected using links, building this way a so called storyboard that gives an idea about the navigation structure.

- **Use Case Modeling** is a technique which was developed to define requirements [16] not for capturing them. However, use cases are sometimes used by companies in the communication between developers and customers because they are an easy understandable graphical representation for costumers and future users of a software application. A use case model consists of actors, use cases and relationships between them [35]. It is used to represent the environment by actors and the scope of the system by use cases (functional requirements). An actor is an external element to the system (e.g. a user, another system) that interacts with the system as a black box. A use case describes the sequence of interactions between the system and its actors when a concrete function is executed. An actor can take part in several use cases and a use case can interact with several actors.

  The main advantage of use case modeling is that a use case model is easy to be understood by the user or the customer as well as by the developers. However, sometimes they are not concrete or detailed enough [37, 15, 31]. Thus, they can be supplemented with textual information or another technique like activity diagrams.

- **Questionnaire and Checklist** is a technique that consists of preparing a document with questions for which only short and concrete answers or even with a limited choice of answers (checklist) is possible. The questionnaire can be completed during an interview or it can be used to get information independently from an interview. The drawback of this technique is that the analyst needs certain knowledge about the problem domain and the application to be built in order to prepare the questionnaire and checklist.

- **Terminology Comparison** is a technique that does not resolve the problem of requirements elicitation on its own. Instead, it is a complementary technique used to overcome the communication difficulties, that may arise among developers and users, who do not use the same language. The comparison is used to get a consensus about the terminology which will be used in the project. Therefore, it is necessary to identify those words used for the same concept (correspondence), similar words to express different concepts (conflicts) or if there is not exact concordance in the vocabulary or concepts (contrast) [28].

The requirements engineering community has proposed many other techniques to capture requirements, such as the analysis of similar systems or documentation. Nevertheless, we consider that the techniques briefly described above provide a representative set of the most frequently used ones.

2.2 Requirements Specification

For the requirements definition activity, many techniques have also been proposed. In this section, the most widely used are briefly described.

- **Natural Language.** It is an ambiguous technique to define requirements. Requirements are described in natural language without any kind of rules. Although this procedure is often criticized, it is quite often used in practice.

- **Glossary and Ontology** are used to define the terminology that should be used in every software project where stakeholders with different background work together. This aspect is critical in the development of Web applications as the development team is usually an interdisciplinary one [19]. Therefore, many methodologies propose the use of a glossary in order to define and maintain the most important and critical concepts related to the application.

  If an ontology is defined, it means that in addition to concepts, the relationships between these concepts are specified. None of the methodologies for the development of Web applications described in this work propose the use of ontologies. Thus, we did not include this technique in Table 2.

- **Templates.** They are used to describe the objectives and requirements using natural language, but in a structured way. A template is a table whose fields have a predefined structure and are filled in by the development team using the user's terminology. Templates – also known as patterns – are less ambiguous than descriptions in natural language due to their structure. However, if templates are too structured they could be difficult to fill and maintain.

- **Scenarios** consist of the description of the characteristic of the application by means of a sequence of steps [22]. Scenarios can be represented in different ways: as texts or in a graphical form, e.g. by use cases [38]. The analysis of such scenarios provides important information about the requirements of the application [24]. Scenario notations are integrated in many object-oriented analysis techniques.

- **Use Case Modeling** has been widely accepted as a technique to define requirements although it is also used in requirements eliciting as described in the previous section. However, it has the disadvantage that it is ambiguous when defining complex requirements [37, 15, 31]. For this reason, some approaches that define use cases propose to add a textual description using templates or a more detailed diagrammatic representation [19, 37].

- **Formal description** is another important group of techniques that proposes in contrast to natural descriptions the use of formal languages to specify requirements. Algebraic specifications for example, have been applied in software engineering for some years. However, they are difficult to be used and understood by customers. Its main disadvantage is that they do not facilitate the communication between customer and analyst. Conversely, it is the least ambiguous requirements representation allowing for automatic verification techniques.

- **Prototypes** are a valuable tool for providing a context within which users are able to better understand the system they want to build. There is a wide variety of prototypes that range from mock-ups of screen designs to test versions of software products. There is a strong overlap with the use of prototypes for validation.

2.3 Requirements Validation

Once requirements are defined, they have to be validated. Through requirements validation the requirements specification is checked to correspond to the user's needs and the customer's requirements [24]. Only few approaches provide techniques to validate requirements. Most of them only define some guidelines about how developers and customers should review the requirements specification in order to find inconsistencies and mistakes. The following is an overview of techniques that are appropriate for requirements validation:

- **Review or Walk-through** is a technique which consists in reading and correcting the requirements definition documentation and models. Such a technique only validates the good interpretation of the information. The verification of documentation inconsistencies and the detection of missing information require more sophisticated methods.

- **Audit** consists of a check of the results presented in the review documentation. The results are compared with a checklist predefined at the start of the process. It provides only a partial review of the information and results.

- **Traceability Matrix** consists of a comparison of the application objectives with the requirements of the system [8]. A correspondence is established between objectives and how they are covered by each requirement. This way, inconsistencies and non-covered objectives will be detected.

- **Prototyping for validation** is a technique that consists in building tools based on the requirements specification, i.e. the developers' interpretation of the systems requirements. These prototypes usually only implement a partial set of functional requirements but provide a global vision of the user interface [27]. In order to use this technique the user has to understand that what he is observing is only a prototype and it is not the final system.

3 Requirements Engineering in current Web Methodologies

The development of Web applications has several characteristics that differ from the development of other kinds of applications. On the one hand, many different kinds of stakeholders participate in the development process: analysts, customers, users, graphical designers, marketing, multimedia and security experts, etc. On the other hand, the main features of these systems are the navigational structure, the user interface and the personalization capability. The structure requires an intuitive guide to avoid that the user "gets lost in the navigational space" [27]. The design of the user interface often

has to take into account multimedia and marketing aspects. These special design aspects not only have to be handled differently during design, but already be considered during the requirements specification [9].

In this chapter we give an overview of those Web approaches which propose specific techniques or models to deal with requirements. Of course, there are more Web methodologies in use which were not included in this survey. This study is focused on requirements, thus we describe mainly the requirement phase of each approach.

Most of the methodologies analyzed and compared in this work provide a classification of requirements. However, the terminology used in these methodologies is not always the same. In order to make the description of each methodology comparable to the others, a general classification of requirements for Web applications is shown previously to the outline of the methods. It is based on the state of the art of Web methodologies.

- *Functional requirements* are capabilities that a system must exhibit in order to solve a problem. Functional requirements can be sub-classified in:
  - *Data requirements* also known as conceptual requirements, content requirements or storage requirements. These requirements establish how information is stored and administrated by the application.
  - *Interface requirements (to the user)* also known as interaction requirements or user's requirements. They give an answer to how the user is going to interact with the Web application.
  - *Navigational requirements* represent users' navigation needs through the hyperspace.
  - *Personalization requirements* also known as customization or adaptation requirements. They describe how a Web application has to (dynamically) adapt itself, depending on the user or environment profile.
  - *Transactional requirements,* also known as internal functional requirements or service requirements, express what the Web application has to compute internally, without considering interface and interaction aspects.
- *Non-functional requirements* act to constraint the solution, e.g. portability requirements; reuse requirements, usability requirements, availability requirements, performance requirements, etc.

In this section, we only include those Web proposals which contain the phase of the requirements handling in the life cycle of their development process. Some of them covered the requirements phase in early versions; others included it only after a revision. Methodologies are outlined chronologically according to their first publication that included requirements specification. The chronological arrangement gives us an idea of how requirements engineering for Web applications has evolved.

3.1 WSDM: Web Site Design Method

WSDM is a user-centered approach for the development of Web sites that models the application based on the information requirements of the users' groups [7]. Its development process is divided into four phases:

- *User modeling*, where users are classified and grouped in order to study system requirements according to each user group,

- *Conceptual design*, where a class diagram is designed to represent the static model of the system and a navigational model to represent the possibilities of navigation,
- *Implementation design*, where models of the conceptual design are translated into an abstract language easily to be understood by the computer, and
- *Implementation,* where the implementation design result is written in a specific computer language.

We focus on the user modeling phase, which is the relevant one for this work. It aims on the identification of the different users' roles by performing the following two tasks:

- *Users' classification* is the identification of the potentials users/visitors of the Web site and their classification according to their interests and navigation preferences. WSDM proposes to analyze the organization environment where the application will be used, and centers the attention on the stakeholders of the business processes supported by the application. In WSDM the relationships between stakeholders and the business process activities performed are graphically represented by conceptual maps of roles and activities.
- *Users' group description* is the detailed description of the users' groups identified in the previous task. The information requirements, functional requirements and security requirements for each user's group are described with the help of a data dictionary.

The remaining phases in the WSDM process are based on the users' classification of this first phase.

3.2 SOHDM: Scenario-based Object-Oriented Hypermedia Design Methodology

The SOHDM approach [21] was the first approach stressing the importance of a process that allows the analysts to capture and define the applications requirements. SOHDM has similarities with OOHDM [34] among others, but it proposes a requirement specification based on scenarios.

The following six tasks are performed during the life cycle of SOHDM; for this work, only the first one is relevant:

- *Analysis*, where requirements are describe using scenarios;
- *Object model realization*, where a class diagram is built in order to present the static structure of the system;
- *View design*, which expresses how the system will be presented to the user;
- *Navigational design*, where a navigational class model is developed in order to express the possibilities of navigation in the system;
- *Realization of the implementation*, where Web pages, the interface and also the database are developed; and, finally,
- *Construction of the system*, where the system is built.

The requirements definition starts on designing a so called context diagram, similar to the data flow diagrams (DFD) defined by Yourdon [40]. To build such a context diagram the analyst has to identify the external entities that communicate with the application, and the events that trigger the communication between these entities and the application. The set of events is specified as a table

showing the entities that participate in an event. SOHDM proposes to associate a scenario with each event. Scenarios are graphically represented using a proprietary notation called SAC (Scenario Activity Chart). A scenario describes the interaction process between the user and the application when an event triggers an activity. It specifies the activity flow, objects involved and transactions performed.

SOHDM proposes a process to get the conceptual model of the application out of these scenarios. The proposed conceptual model is represented by a class diagram. The next step in the SOHDM development process is the regrouping of these classes with the objective to obtain a navigational class diagram.

3.3 RNA: Relationship-Navigational Analysis

RNA [3] is a methodology that offers a sequence of steps to develop Web applications focusing mainly on analysis. Its phases are:

- *Phase 1 - Environment analysis:* the objective is to analyze the audience's characteristics. Stakeholders of the application are identified and classified in different groups according to their roles (similar to the user modeling phase of WSDM).
- *Phase 2 – Element analysis:* in this phase all elements that are of interest to the application are identified, e.g. documents, forms, information, mock-ups, etc.
- *Phase 3 – Meta-knowledge analysis:* achieves to build a schema of the application. RNA proposes to identify objectives, processes and operations related to the application, and to describe the relationships between those elements.
- *Phase 4 - Navigation analysis:* in this phase, the schema of the previous one is enlarged with navigation features.
- *Phase 5 – Implementation analysis*: consists of the identification of how the models described in phase 4 will be produced in a computable language.

RNA only provides some guidelines of the actions to be performed in each phase. Neither modeling concepts nor a notation is proposed, but the RNA approach is one of the methodologies that first focused on the importance of requirements specification in the development process of Web applications. It emphasized the need of the separation between the analysis of conceptual requirements and the analysis of navigational requirements.

3.4 HFPM: Hypermedia Flexible Process Modeling

The Hypermedia Flexible Process Modeling (HFPM) presented by Olsina [26] is a wide engineering-based approach, which includes analysis-oriented *descriptive* and *prescriptive* process modeling strategies. It includes technical, management, cognitive and participatory tasks. Therefore, HFPM provides guidelines for the planning and managing of a Web project covering the whole life cycle of such a software project. It consists of thirteen phases; for each phase HFPM defines a set of tasks. For the purpose of this work, the most relevant is the requirements model whose related tasks are defined as follows:

- *Problem description.* HFPM does not prescribe a concrete technique to perform the problem description, e.g. natural language can be used.
- *Description of functional requirements* using use cases.

- *Data modeling* for the identified use cases. It proposes the design of a class diagram.
- *User interface modeling* using sketches and prototypes to be used in the presentation of drafts to the customer.
- *Non-functional requirements description*, such as security, performance, etc.

HFPM proposes on the one hand a detailed process to handle requirements. On the other hand it does not prescribe specific techniques, which can be chosen freely by analysts and developers.

3.5 OOHDM: Object Oriented Hypermedia Design Model

OOHDM is a widely accepted method for the development of Web applications [34], whose first versions focused on design and did not include requirements engineering. The process in OOHDM is divided in four phases producing the following results:

- The *conceptual model*, represented as a class model, is built in order to show the static aspect of the system.
- The *navigational model* consists of a navigation class diagram and a navigation structure diagram. The first one represents the static possibilities of navigation in the system. The second one extends the navigation class diagram including access structures and navigation contexts.
- The *abstract interface model* is developed using a special technique named ADVs [34].
- The *implementation* consists in the implemented code and is based on the previous models.

The capture and definition of requirements were introduced later in OOHDM by Vilain, Schwabe and Sieckenius [37], proposing the use of user interaction diagrams (UIDs). UIDs base on the well known technique of use cases. Use cases are used to capture the requirements but are considered in OOHDM as ambiguous and insufficient for the definition of the requirements that Web applications have, mainly related to the interaction between the user and the system. Therefore, for the specification of the requirements, this approach suggests the refinement of use cases building UIDs, which are used to graphically model the interaction between users and system without considering specific aspects of the interface. The process to get an UID from a use case is described very carefully in the approach.

3.6 UWE: UML-based Web Engineering

UML-based Web Engineering (UWE) is a methodological approach for the development of Web applications based on the Unified Process [17, 5]. It is based mainly on the most relevant concepts provided by other methods, but defines a UML notation (UML profile), sticks to the diagrammatic techniques proposed by the UML and defines a systematic and semi-automatic design process [14].

UWE covers the whole life cycle of Web applications and focuses on adaptive applications. It includes a specific requirements engineering phase where requirements elicitation, specification and validation are handled as separate activities of the process. The final result of the requirements capture in UWE is a use case model completed with documentation describing the users of the application, the adaptation rules, the interfaces and the details of the use case relevant for the use case implementation, which can be described textually or modeled by UML activity diagrams.

UWE classifies requirements into two groups: functional and non-functional. Functional requirements contemplated in UWE are:

- Content requirements
- Structure requirements
- Presentation requirements
- Adaptation requirements
- User model requirements

Moreover, UWE proposes interviews, questionnaires and checklists as appropriated techniques for the requirements capture, and use cases, scenarios and glossaries for the requirements specification. To validate them, UWE proposes walk-through, audits and prototypes [19].

3.7 W2000

W2000 [1] is an approach that also extends UML notation to model multimedia elements. These multimedia elements are inherited from HDM (Hypermedia Design Model) [12]. The development process of W2000 is divided into three phases: requirements analysis, hypermedia design and functional design. The first one is the most interesting for our survey.

The requirements analysis in W2000 is divided into two sub-activities: functional requirements analysis and navigational requirements analysis. The requirements elicitation starts with an analysis of the different user roles, i.e. the actors which will interact with the application. Every identified actor has his own navigation and functional requirements model. The latter model is represented by a UML use case model. The navigational requirements are modeled in another use case diagram representing the navigation possibilities of the actors. The graphic notation is defined as a UML extension.

3.8 WebML: Web Modeling Language

The Web Modeling Language (WebML) is a high-level specification language for hypermedia applications. WebML follows the style of both, Entity-Relationship and UML offering a proprietary notation and a graphical representation using the UML syntax. This notation is complemented with a set of activities to be performed for the development of Web applications, such as requirements specification, data design and hypertext design [6].

The methodology focuses on requirements collection and requirements specification. It proposes the use of techniques, such as interviewing and analysis of documentation, but retrains from the use of prescriptive checklists for requirements capture. Requirements collection starts with user identification and personalization needs. In addition data requirements and functional as well non-functional requirements are gathered. To note is that navigation or specific hypertext structuring requirements are not treated separately.

Requirements specification (called requirements analysis) consists in a classical use case specification supplemented with a semi-structured textual description. The use of activity diagrams is proposed by this method to express the workflow of complex use cases. A template based description and mock-ups (sketches) are suggested for the specification of the site view and the style guidelines. Finally, acceptance tests are proposed mainly to check non-functional requirements.

3.9 NDT - Navigational Development Techniques

NDT (Navigational Development Techniques) [10] is a technique to specify and analyze the navigation aspects in Web applications. NDT focuses on the elicitation and specification techniques selected by NDT for the capture and definition of requirements. The requirements analysis workflow in NDT starts capturing requirements and studying the environment applying interviews, brainstorming and JAD techniques. In a second step the system objectives are captured and described. Based on these objectives the system requirements are identified; NDT classifies them into:

- Storage information requirements
- Actor requirements
- Functional requirements
- Interaction requirements
- Non-functional requirements

Interaction requirements are represented by phrases and visualization prototypes. Phrases show how the information of the system is retrieved and are represented by a special language named BNL (Bounded Natural Language) [4]. Visualization prototypes are used to represent the system navigation, data visualization and user's interaction.

The whole process to elicit and specify objectives and requirements proposed by NDT is mainly based on templates or patterns. In addition, it uses other requirements definition techniques like use cases and glossaries. The NDT approach proposes a different template for each kind of requirement, so requirements and objectives are described in a structured way. Some fields in the templates only accept specific values allowing for a systematic process. The requirements specification workflow finishes with the revision of the requirements catalogue and the development of a trazability matrix which makes the evaluation of whether the specification covers all the possible requirements.

In the context of the NDT project a case tool, named NDT-Tool, has been developed. This tool supports the filling of the templates and automatic extraction of the design results out of the templates.

3.10 Design-driven Requirements Elicitation

The Design-driven Requirements Elicitation is a part of the design-driven process proposed by Lowe and Eklund [25] in order to develop Web applications. It consists of capturing, defining and validating requirements during the design process, i.e. the design activities should be carried out in such a way that the requirements could be handled and managed at the same time. The process is based on prototyping in order to explore possible solutions and problems to be solved. Users and customers define the requirements based on the study of these prototypes. It is an iterative process, which consists of reducing customers and clients' doubts. The cycle has three phases: evaluation, specification and construction.

This design-driven process was defined based on an exhaustive analysis of "best practices" in the development of Web commercial applications. It treats all the requirements in the same manner. The requirements are: content, interface protocol, navigational structure, look and feel, data internal representation, versions, change control, security, content management, control access, efficiency, user

monitoring, functionality support, system adaptation, user identification, etc. In the comparison tables of the next section we use the short form DDDP for the design-driven development process[b].

4 Comparative Study

We have based our comparative study on three main aspects. The first one is the analysis of the types of requirements handled by each methodology. The second aspect is the study of the techniques employed and the phases covered in each approach. The last one evaluates the degree of detail of each approach in terms of its development process, the applied techniques and the results produced. Finally, some other aspects are outlined.

4.1 Types of Requirements

Using the classification introduced at the beginning of section 3, the first objective of this comparison was to establish which types of requirements are treated by each approach. Table 1 shows these results for the methodologies briefly described in sections 3.1 to 3.11 and the six types of requirements: data, user interface, navigation, adaptive, transactional and non-functional.

Approaches are ordered chronologically, what allows us to observe the evolution of the requirements engineering relevance in those methodologies. The first approaches focused mainly on data and user interface requirements. More recently some methodologies have been developed or already existing ones have been extended to manage adaptive, navigation and transactional requirements. The idea of separation of concerns was since the beginning a characteristic of almost all Web methodologies, like HDM [12], OOHDM [34], etc. However, this separation of concerns was only applied to the design and implementation phases of the development process. Nowadays we can observe a clear tendency towards a separation of concerns from the very beginning, i.e. already during the requirements elicitation phase. It is interesting to remark, that the use of different terminology for the same or similar concepts made a comparison study difficult. We stress the need to standardize the terminology used in Web methodologies.

	Data Req.	User Interface Req.	Navigational Req.	Adaptive Req.	Transactional Req.	Non-Functional Req.
WSDM	✓			✓		✓
SOHDM	✓	✓			✓	
RNA	✓	✓	✓		✓	
HFPM	✓	✓	✓			✓
OOHDM	✓	✓	✓			
UWE	✓	✓	✓	✓		✓
W2000			✓	✓	✓	
WebML	✓	✓		✓		✓
NDT	✓	✓	✓	✓	✓	✓
DDDP	✓	✓	✓	✓	✓	✓

Table 1: Requirements Handled by Each Approach

[b] The short form DDDP is not used by the authors of the approach.

4.2 Activities and Techniques

The following table shows the differences regarding the techniques that are used by each methodology in each phase, i.e. during the activities of elicitation; specification and validation (see section 2). If a methodology proposes a non-standard technique or a specific technique it is explicitly indicated.

		WSDM	SOHDM	RNA	HFPM	OOHDM	UWE	W2000	WEbML	NDT	DDDP
Capture	Interviewing	✓		✓			✓		✓	✓	✓
	JAD									✓	
	Brainstorming									✓	
	Concept Mapping	Role-Activity									
	Use Cases Modeling					✓					
	Questionnaire/Checklist						✓				
	Sketching & Storyboarding								✓		
	Other Techniques		DFD						Document analysis		
Definition	Natural Language	✓		✓	✓				✓		
	Glossaries				✓		✓			✓	
	Templates/Patterns								✓	✓	
	Scenarios		SAC				✓				
	Use Cases Analysis				✓	✓	✓	✓	✓	✓	
	Formal Language										
	Prototyping										✓
	Other techniques		Event List		Interface Sketches	UIDs				BNL Phrases	
Validation	Review/Walk-through						✓			✓	
	Audit						✓				
	Matrix of trazability									✓	
	Prototyping				✓		✓				✓
	Other techniques								Acceptance Tests		

Table 2: Techniques used in the Capture, Definition and Validation Phases

Several conclusions can be obtained from this table. It is possible to indicate that interviewing is the most popular technique during requirements capture. Similarly requirements specification with use cases is the winner for the definition of requirements. We observe that many methodologies handle the capture as part of the definition activity.

In this table we also observe that Web methodologies focus on the requirements definition activity. During this activity, the use case technique (the most used one) is applied in different ways. Some methodologies, such as HFPM, apply the original use case technique. However, other approaches, like OOHDM, NDT or UWE, believe that use cases are ambiguous or insufficient for the specification, and so complement this technique with more concrete models, such as UIDs, templates or UML activity diagrams, respectively. This more detailed specification helps for a more systematic development.

Most of the methodologies consider validation not as relevant as the other two phases: capturing and specification. The validation techniques proposed mainly focus on reviewing the requirements models or the textual descriptions of the requirements. Some of the approaches analyzed do not even include the validation phase in their requirements engineering process.

4.3 Degree of Detail

Another perspective under which a comparative study can be carried out involves the way how requirements engineering approaches are defined. Some methodologies concentrate largely on the development process, others focus on the techniques or on the structure of the results that must be produced. Therefore, we classify the approaches in three categories:

- *process-oriented*, that is, if the approach describes the steps of a process to be followed in order to perform the requirements capture, definition and validation;
- *technique-oriented*, that is, if it describes the techniques to be applied during the process;
- *product-oriented*, that is, if it gives a description of the results which must be produced during the process;

We analyzed the definition of the approaches and evaluated how detailed they are in the description of the process, the techniques and the products. The result of this evaluation is shown in table 3. The evaluation is done separately for each phase of the requirements engineering selecting one value as follows:

- *process-oriented:* the approach clearly describes the steps to follow (+), the process without details (o), or does not indicate any process at all (-)
- *technique-oriented:* the approach clearly depicts the techniques and the way to apply them (+), it enumerates the techniques to apply (o), or it does neither propose any concrete technique nor references any general techniques (-)
- *product-oriented*: the approach clearly describes the structure of the product to be produced (+), it describes the product without detailing its structure (o), or it does not give any indication about the resulting product (-)

	Process-oriented	Technique-oriented	Product-oriented
WSDM	O	-	-
SOHDM	-	+	-
RNA	+	-	-
HFPM	+	O	+
OOHDM	O	+	-
UWE	+	O	O
W2000	O	O	-
WebML	O	O	+
NDT	O	+	+
DDDP	+	O	-

Table 3: Degree of Detail in Processes, Techniques and Products

The values listed in table 3 can be grouped and represented schematically as shown in figure 2. This graphical representation has led us to assess that current methodologies mainly focus on the process. An extreme example of this fact is RNA, which only describes the process without mentioning techniques or the layout of the results.

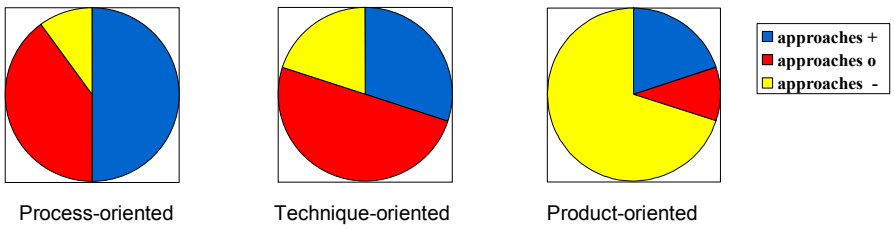

Figure 2: Graphical Representation of the Degree of Detail in Processes, Techniques and Results

Another important conclusion is the little importance given to the result produced during the requirements engineering process. Many of the approaches only enumerate the models to be produced without indicating a structure of the documents. Only HFPM and NDT describe with details how the requirements documentation should be structured. Such templates are very useful to support the development team in the documentation process, but certain flexibility is required for the tailoring of such documents to needs, formats or restrictions of a specific project environment.

4.4 Other aspects

There are some other aspects that can be used to compare the different requirements engineering approaches, which we want to outline briefly in this section.

One of the main pillars of good requirements engineering practices is the use of techniques that support fluent communication between users and technical software developers. The requirement

engineer is responsible for communication and acts as mediator. Good communication is an aspect that is difficult to measure and therefore difficult to compare.

It is important to highlight the two trends for representation used in requirements engineering: graphical and textual representation. Extreme examples are OOHDM and UWE, which performs requirements capture and definition using visualization, and conversely, approaches such SOHDM and NDT based on textual representation. In addition is it worth mentioning, that a similar technique may be differently used or represented by different methodologies, e.g. scenarios in UWE and SOHDM.

Another aspect that is worth being compared is the tool support. We have detected a lack of CASE tool support. Only few methods suggest the use of tools. Even less approaches propose a specific tool for the requirements specification. Such is the case of the tool developed within the UWA project [36] and the NDT-Tool for the requirements definition following the NDT method.

5 Conclusions

In this work we have presented the state of the art of requirements engineering in methodologies used for the development of Web applications. To achieve this purpose, we started describing the structure of the requirements engineering process and the most common techniques used in such a process in the classic software development for non-Web applications. The process includes three main activities: capture, definition and validation of requirements. The techniques most frequently used to perform these activities are among others use cases, scenarios, sketches and storyboards, questionnaires and checklists, reviews and walk-troughs, prototyping, etc.

In a second step we gave an outline of the methodologies for the Web describing how these approaches cover the aspects related to requirements engineering. Finally, the approaches are compared from different points of view, such as types of requirements handled, activities covered and techniques used, orientation and depth of the applied techniques.

As the main result of our study we can claim that there is still a great research potential in the field of requirements engineering for Web applications. In contrast to non-Web specific methodologies, as shown in other comparative studies [18, 2, 9, 32], the Web approaches we analyze in this survey focus on design aspects and do not centre its attention on requirements engineering, although the risks of an incomplete or insufficient requirements definition and validation is well-known.

Some Web methodologies assume that classical requirements engineering techniques can be used in Web engineering. In effect, some techniques, such as use cases, interviewing and checklists are suitable for the Web environment. However, Web systems have some specific characteristics that make the requirements process very critical. The high number of stakeholders, the navigation aspect, the layout quality required for the interface and multimedia synchronization are only some of the Web characteristics that have to be specially treated during the phase of requirements. Specific requirements engineering should be included in the Web methodological proposals in order to offer a good reference guide to the development team.

We hope the results presented in this article will help Web developers to select the appropriate requirements engineering techniques and include them in the development process of Web applications. In addition, it should help in the continuous improvement process of the existing Web methodologies to focus more on requirements engineering, and therefore contribute to improve the quality of the Web applications that are built using these methodologies.

Acknowledgements

We would like to thank Martin Wirsing and Andreas Kraus of the Ludwig-Maximilians University of Munich, Manuel Mejías, Jesús Torres and Miguel Toro of the University of Seville and Cristina Cachero of the University of Alicante for helpful suggestions and comments.

References

1 Baresi L., Garzotto F., Paolini P (2001). *Extending UML for Modeling Web Applications*. In proceedings of the 34th Annual Hawaii International Conference on System Science. IEEE Computer Society.

2 Barry, C. & Lang, M. (2001) *A Survey of Multimedia and Web Development Techniques and Methodology Usage.* IEEE Multimedia. April-June 2001, 52-60.

3 Bieber M., Galnares, R., Lu, Q. (1998). *Web Engineering and Flexible Hypermedia.* The Second Workshop on Adaptive Hypertext and Hypermedia, Hypertext´98, Pittsburg, USA.

4 Brisaboa, N. R., Penabad, M. R., Places, A. S., Rodríguez, F. J. (2001). *A Documental Database Query Language*. String Processing and Information Retrieval -SPIRE 2001.

5 Booch G., Rumbaugh, J., Jacobson, I. (1999). *Unified Modeling Language User Gu*ide. Addison-Wesley.

6 Ceri, S. Fraternali, P., Bongio, A., Brambilla M., Comai S., Matera M. (2003). *Designing Data-Intensive Web Applications.*Morgan Kaufman.

7 De Troyer, O., Leune, C. (1997). *WSDM: A User Centered Design Method for Web Sites*. Technical Report of Tilburg University, Infolab. Belgium.

8 Durán A., Bernárdez, B., Ruiz, A., Toro M. (1999). *A Requirements Elicitation Approach Based in Templates and Patterns*. Workshop de Engenharia de Requisitos. Buenos Aires, Argentina.

9 Escalona, M.J., Mejías, M., Torres, J. (2002). *Methodologies to develop Web Information Systems and Comparative Analysis***.** Informatik/Informatique. núm. 2/2002 de I/I.

10 Escalona, M.J., Torres, J., Mejías, M. (2002). *Requirements Capture Workflow in Global Information Systems.* Proceedings of OOIS. Springer-Verlag. Montpellier, France.

11 Ferreira, M.J., Loucopoulos, P. (2001). *Organisation of Analysis Patterns for effective Re-use.* Proceedings of the International Conference on Enterprise Information Systems. ICEIS 2001. Setubal, Portugal.

12 Garzoto F., Schwabe D. and Paolini P. (1993) *HDM-A Model Based Approach to Hypermedia Aplication Design*. ACM Transactions on Information System, 11 (1), pp 1-26.

13 Grünbacher P. (2003) *Requirements Engineering for Web Applications*. In Web Engineering, Kappel G., Pröll B., Reich S., Retschitzeger W. (Eds.), dpunkt verlag (in German).

14 Hennicker, R., Koch, N. (2000). *A UML-based Methodology for Hypermedia Design.* Lecture Notes in Computer Science. Proc. UML'2000. York, England.

15 Insfrán, E., Pastor, O., Wieringa, R. (2002). *Requirements Engineering-Based Conceptual Modeling.* Requirements Engineering Journal, 7(2):61-72, 2002.

16 Jacobson, I. (1995). *Modeling with Use Cases: Formalizing Use Case Modeling.* Journal of Object-Oriented Programming,

17 Jacobson I., Booch G., Rumbaugh J. (1999). *The Unified Software Development Process.* Addison Wesley.

18 Koch, N. (1999). *A Comparative Study of Methods for Hypermedia Development.* Technical Report 9905. Ludwig-Maximilian-University, Munich, Germany.

19 Koch, N. (2001). *Software Engineering for Adaptive Hypermedia Applications*. Ph. Thesis, FAST Reihe Softwaretechnik Vol(12), Uni-Druck, Munich, Germany

20 Kruchten, P. (1998). *The Rational Unified Process*. Addison Wesley

21 Lee, H., Lee, C., Yoo, C. (1998). *A Scenario-based Object-oriented Methodology for Developing Hypermedia Information Systems.* Proceedings of 31st Annual Conference on Systems Science. Sprague R.

22 Liu, L., Yu, E. (2001). *From Requirements to Architectural Design using Goals and Scenarios* Proceedings of the 6th Micon Workshop. Canada.

23 Livesey D., Guinane T. (1997). *Developing Object-Oriented Software, An Experience-Based Approach* (IBM's OOTC), Prentice Hall

24 Lowe, D., Hall, W. (1999). *Hypermedia and the Web. An Engineering approach.* John Wiley & Son.

25 Lowe D., Eklund J. (2002). *Client Needs and the Design Process in Web Projects.* Web Engineering Track of the WWW2002 Conference.

26 Olsina, L. (1998). *Building a Web-based Information System applying the Hypermedia Flexible Process Modeling Strategy*. 1st International Workshop on Hypermedia Development, Hypertext´98, Pittsburg, USA.

27 Olsina, L. (1999). *Metodología Cualitativa para la Evaluación y Comparación de la Calidad de Sitios Web.* Ph. Tesis. Facultad de Ciencias Exactas. Universidad de la Pampa. Argentina.

28 Pan, D., Zhu, D., Johnson, K. (2001). *Requirements Engineering Techniques*. Internal Report. Department of Computer Science. University of Calgary. Canada.

29 Pastor, O., Insfran, E., Pelechano, V., Romero, J., Meseguer, J. (1997). *OO-METHOD: An OO Software Production Environment Combining Conventional and Forma Methods*. CAiSE'97. International Conference on Advanced Information Systems.

30 Raghavan, S., Zelesnik, Ford, G. (1994). *Lectures Notes of Requirements Elicitation. Educational Materials* CMU/SEI-94-EM-10.

31 Regnell B., Kimbler K., Wesslen A (1995). *Improving the Use Case Driven Approach to Requirements Engineering*. 2nd IEEE International Symposium on Requirements Engineering. IEEE. York, UK.

32 Retschitzegger, W., Schwinger, W. (2000). *Towards Modeling of Data Web Applications - A Requirements Perspective*. Proceedings of the American Conference on Informating Systems AMCIS 2000, Vol 1, 149-155.

33 Sawyer P., Kotonya G. (2001). *Software Requirements*. Chapter 2 of the IEEE SWEBok Project Report.

34 Schwabe D., Rossi G. (1998). *Developing Hypermedia Applications using OOHDM*. Workshop on Hypermedia Development Process, Methods and Models, Hypertext´98, Pittsburg, USA.

35 UML (2003). *Unified Modeling Language*. Version 1.5. www.omg.org

36 UWA (2001), *UWA Requirements Elicitation: Model, Notation, and Tool Architecture*. www.uwaproject.org

37 Vilain, P., Schwabe, D., Sieckenius, C. (2000). *A diagrammatic Tool for Representing User Interaction in UML*. Lecture Notes in Computer Science. Proc. UML'2000. York, England.

38 Weidenhaupt, K., Pohl, K., Jarke, M., Haumer, P. (1999). *Scenarios in Systems Development: Current Practice.* IEEE Software. 2, 34-45.

39 Wieringa. , R.J. (2001) Software requirements engineering: The need for systems engineering and literacy. Requirements Engineering Journal, 6(2):132-134, 2001.

40 Yourdon E (1989). *Modern Structured Analysis*. Prentice-Hall.