Advances in Engineering Software 40 (2009) 1306–1317

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Integrating usability requirements that can be evaluated in design time into Model Driven Engineering of Web Information Systems ☆

Fernando Molina *, Ambrosio Toval

Software Engineering Research Group, Department of Informatics and Systems, University of Murcia, Facultad de Informática, Campus de Espinardo, 30100 Murcia, Spain

A R T I C L E I N F O

Article history:
Received 22 September 2008
Received in revised form 19 November 2008
Accepted 19 January 2009
Available online 10 April 2009

Keywords:
Web engineering
Requirements metamodelling
Usability evaluation
Software quality

A B S T R A C T

In recent years, Web Engineering development projects have grown increasingly complex for and critical to the smooth running of organizations. However, recent studies reveal that a high percentage of these projects fail to attain the quality parameters required by stakeholders. The inadequate consideration of requirements management activities together with the absence of attention to the elicitation and evaluation of requirements and metrics related to certain quality attributes which are of special importance in this kind of systems, such as usability, have proved to be some of the main causes of this failure. This paper attempts to reduce some of the quality failures detected in Web Engineering development projects by proposing the consideration and evaluation of quality attributes from early stages of the development process. The presented approach therefore commences with a reinforcement of the requirements related activities in this discipline, which is carried out by using a requirements metamodel. Once these requirements have been identified, the approach focuses on the extension of the conceptual models used by Web Engineering methodologies with the aim of allowing the explicit consideration of usability requirements along with the evaluation of quality metrics during the design of the system. An example of an application illustrating how the approach can be used, along with the automatic support which was developed for it, are also shown.

1. Introduction

The development of Web Information Systems (WIS) has undergone an exponential growth in the last decade. Initially, these systems were used only as a means to disseminate information. However, their complexity has recently increased, they are present in numerous domains being used by millions of users around the world and they have become critical systems for the business strategies of many organizations [36]. This growth, together with some particular features that make WIS development projects different from other software developments [36], have compelled the organizations to adapt their software development processes to deal with the idiosyncrasy of WIS [35]. Moreover, as a result of its awareness of such difficulties, the WIS research community has developed numerous techniques, tools and methodologies within the scope of a new discipline, called Web Engineering (WE), which promotes the establishment and use of sound scientific, engineering and management principles, and disciplined and systematic approaches for the development, deployment and maintenance of web-based systems [19].

However, as various surveys and research publications reveal [31,15], the development of this kind of systems is not exempt from errors and the WIS finally developed, regardless of their scope (business, engineering, scientific, etc.), do not always satisfy the quality requirements demanded by their users. These studies highlight that the top five problem areas of large-scale WE projects are (1) failure to meet business needs (84%), (2) project schedule delays (79%), (3) budget overrun (63%), (4) lack of required functionality (53%) and (5) poor quality of deliverables (52%). In fact these problems, far from being new, are quite similar to those encountered in traditional Software Engineering, in which it has already been proved [9,24] that they are often a symptom of an inadequate management of the tasks related to the requirements workflow of the project.

It is therefore surprising that, in spite of this reality, organizations which carry out the WIS projects or WE methodologies currently used in industry (such as e.g. WebML [8], UWE [33] or OO-H [21]), are still not, to the best of our knowledge, paying the necessary amount of attention to requirements management activities in WIS projects. In fact, various surveys [16,17] reveal that WE methodologies are still mostly focused on the design workflow of web applications, while the requirements workflow is, at best, only tangentially tackled. Another survey carried out by Lang and Fitzgerald [35] over 160 organizations that develop web-based

☆ Partially supported by the projects DEDALO (TIN2006-15175-C05-03) and the CALIPSO thematic network (TIN2005-24055-E), from the Spanish Ministry of Science and Technology, and MELISA-GREIS (PAC08-0142-335). The first author is partially funded by the Fundación Séneca (Región de Murcia).

* Corresponding author. Tel.: +34 868 884642/03; fax: +34 868 88 4151.
E-mail addresses: fmolina@um.es (F. Molina), atoval@um.es (A. Toval).

software reveals that 60% of the organizations still consider that one of the main problems in their projects is related to the clarity and stability of requirements.

Furthermore, several authors [36,30] claim that WIS systems require special care when dealing with certain quality attributes (namely usability and accessibility), and tailored usability/accessibility evaluation methods have been proposed for this purpose [25]. Again, organizations and methodologies are ignoring this fact, and continue to rely on informal, ad-hoc quality evaluation and testing activities.

As Blaine and Cleland-Huang [6] notes, the elicitation of quality requirements in current web projects is still mostly implicitly understood by the stakeholders. This fact, together with a validation of this kind of requirements which is more difficult to achieve than that of functional ones [6], is likely to lead to problems with their satisfaction on the delivered products. An analysis of the current WIS development methodologies corroborates these arguments, since they do not offer support in dealing with usability requirements during the development process and they delay this task until the system has been completely developed. The definition of methods for ensuring the usability of web applications is, therefore, one of the current research goals in Web Engineering and Human–Computer Interaction (HCI) [37], and studies such as [30] claim that the consideration and evaluation of usability features should be moved to the early phases of the WIS development lifecycle in order to improve the user satisfaction and decrease maintenance costs [30,3].

In view of this situation, this paper proposes a shift in the way that web development is tackled. This shift is oriented towards a reinforcement of the activities related to the consideration and evaluation of quality attributes such as usability from the early stages of the development process. We therefore provide a requirements metamodel with which to formally define the typical elements that participate in WIS requirements elicitation. This metamodel complements the other metamodels used by WIS methodologies due to their progressive alignment with the model driven approaches [49], emphasizing the importance of requirements elicitation in the quality of software products. Bearing in mind that stakeholders propose certain usability requirements that cannot be expressed over the models used by WE methodologies, the existing WE metamodels have been extended with new entities, attributes and relationships in order to foster the explicit definition and evaluation of usability requirements and metrics during the design of the WIS. The aim of this quality evaluation in design time is to improve models used for WIS development which will later influence the quality of the WIS built from them. The approach focuses on navigational models and is accompanied by a tool that offers automatic support for all the activities of which it is made up.

The integration of these contributions to WIS development processes and methodologies may help to reduce the number of failures detected in WIS development projects. It also provides the benefits related to an adequate requirements management and an early consideration of usability, with the final aim of increasing the quality of the WIS developed and, thus, the stakeholders' satisfaction.

The remainder of the paper is organized as follows. Section 2 presents the approach with which to reinforce requirements related activities in WIS projects. In Section 3, the extension of the navigational models used for WIS development which is carried out to support the definition and evaluation of usability requirements and metrics in early phases of software lifecycle is shown. Section 4 shows the automatic support developed for the approach and, in Section 5, an example that further illustrates the use of the approach is presented. Finally, in Section 6, the main conclusions are drawn and further lines of research are outlined.

2. Reinforcing requirements management activities in WE through a requirements metamodel

The tasks related to requirements are among those which are most influential in the success of a software development project [9]. This is no different in WIS development projects, and the organizations dedicated to their development corroborate this. For example, a set of interviews with organizations that develop web-based systems reported in [36] indicated that, for 76% of the respondents, gathering the right requirements was a critical factor for the success of their projects.

However, requirements management activities remain an open issue in the WE discipline [16] and, moreover, WE methodologies usually ignore that the consideration of quality attributes, such as usability, should start in early stages of the development lifecycle as requirements elicitation and WIS design [29]. The following subsections present an approach which, by using a requirements metamodel as basis, attempts to contribute towards filling this gap, emphasizing the role that RE plays in WE, helping stakeholders in their systematic identification of requirements and paving the way for the consideration of usability requirements from the first phases of WIS development process.

2.1. Requirements metamodelling

2.1.1. Introduction

Throughout its history, intensive research within the RE discipline has motivated the development of a great number of approaches dedicated to dealing with requirements, as is the case of proposals which are goal-oriented [56], aspect-driven [46] or are based on requirements templates [47], to name but a few. On most occasions, these approaches use textual descriptions for the requirements specification, which are often gathered in non-formal models [16] or organized in requirements documents which are not usually formally structured [20]. In recent years, the growing impact of the Model Driven Engineering (MDE) approach [49], with proposals such as the OMG's Model Driven Architecture (MDA) [32] or Microsoft's Software Factories [23], has produced a change in the perspective in which these proposals consider requirements. In them, the entire development process is driven by models and, therefore, the requirements must also be represented as models through the use of a requirements metamodel which formally defines the concepts and relationships involved in the RE process [20].

Meanwhile, and bearing in mind that WE is a specific domain in which MDE can be successfully applied [41], the WE methodologies have been influenced by these approaches. This has motivated the most relevant methodologies (such as WebML, UWE or OO-H) to be aligned with MDE, and the development of WIS has been carried out by using different models and transformations among them.

This alignment of both the WE methodologies and the requirements management techniques with the model driven approaches suggests that the proposals developed with the aim of filling the gap related to requirements management in WIS should also be aligned with the model driven development paradigm [16]. With this trend in mind, the following subsections present our approach for a reinforcement of the requirements management activities in WIS.

2.1.2. Definition, concepts and related work

A requirements metamodel defines the concepts and relationships involved in the RE process in a formal manner. The advantages that a metamodel offers are numerous [32]. First, the metamodel defines both the elements that participate in the requirements management process and the relationships between them in an unambiguous way. Moreover, it offers a formal basis upon which tools for (1) the management of the metamodel elements and (2) the definition of transformation rules from requirements to other elements can be constructed [34].

Several approaches for requirements metamodelling have recently appeared, but a reference model to deal with requirements does not exist [20]. To our knowledge, within the concrete scope of WE, the topic of requirements metamodelling has only been tackled in [16,7]. The requirements metamodel presented in [16] follows the design-oriented approach used by most WE approaches, meaning that the concepts that appear in it are very close to the WIS design (for instance, it includes visualization features or search activities as main constructs in the requirements elicitation activities). The main problem of this lack of high-level expressivity is that it is a generally avowed fact that design requirements are difficult to understand by stakeholders who are not directly related to the design. Such stakeholders require a more abstract way in which to express their own requirements, that is, a way that is closer to the domain under which the application is being developed.

This necessity of capturing higher-level communication goals and user requirements in a stable manner is noted in [7], in which a goal-oriented approach to model web requirements is used. The concept of goal was defined in the i^* framework [57] and it models a high-level objective of one or more stakeholders. The concepts in the i^* framework are very useful in the modelling of user goals, although their generality suggests the need to tailor them to specific domains. Following this trend, [7] uses i^* as a basis, but specializes it in order to design a new requirements metamodel that collects particular WE concepts such as, for instance, a web requirements taxonomy.

As with the proposal of Escalona and Aragón [16], the requirements taxonomy proposed by Bolchini and Paolini [7] is similarly closely related to the way in which WE devised the application design at the time of its proposal, and the authors therefore note the need to improve automatic support and to deal with the concept of requirements reuse.

Other requirements metamodelling proposals which are not specifically focused on web-based systems also exist, such as COMET [4], a requirements modelling method that includes a requirements metamodel. However, this metamodel does not pay attention to non-functional requirements and it does not cover any method for requirements validation. REMM [54] presents another requirements metamodel but it does not consider the possibility of capturing the aforementioned high-level communication goals, nor does it cover methods for requirements validation. The concept of requirements reuse presented in [54] is, nevertheless, very useful and it can be adapted to the WIS scope.

In other cases, and since it is difficult to represent all the features included in the different approaches in a single metamodel, new requirements metamodels have been obtained by unifying the common concepts from the other proposals. Goknil et al. [20] follows this trend. However, the proposed metamodel does not consider important concepts such as that of requirements reuse and neither does it contain an automatic support to deal with the metamodel concepts.

To sum up, the existing requirements metamodels present interesting concepts, but there is a need for a metamodel which is independent of particular WE views, quality models or implementation technologies. We have therefore developed a requirements metamodel that synthesizes and simplifies the, from our point of view, most relevant concepts included in well-known RE proposals. Such simplification was necessary to avoid the burden of work usually added by more exhaustive RE practices, on the premise that, in the WE community, baselines must be generated very quickly [35] and therefore straightforward methods through which to gather requirements and connect them with validation methods are needed. This metamodel stresses the importance of performing requirements management activities as a first-order workflow for web methodologies and paves the way towards considering usability requirements from the first stages of the WIS development process. This metamodel is presented in Fig. 1 and is explained in more detail in the following subsection.

2.2. A requirements metamodel for WIS

Fig. 1 shows our approach for requirements metamodelling, which had as its origin an initial contribution to requirements metamodelling for WIS projects [40]. Let us now look at the concepts presented in Fig. 1 in greater depth. The key element in this metamodel is that of `requirement` which can be described by using a set of attributes (hidden in the figure) such as an `identifier`, its `type` and `priority` and its textual `description`. In order to avoid, as far as possible, the ambiguity inherent in natural language, the description of requirements can use `terms` included in a `glossary`. In Fig. 1, requirements can be classified as either `functional requirements` (what the system must do) and `non-functional requirements` (how the system must do it). This classification is useful because, while functional requirements usually rely on `test cases` for their validation, non-functional

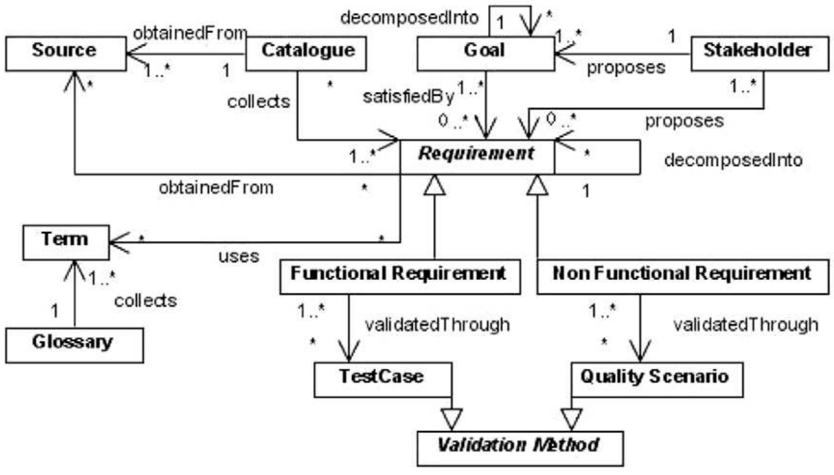


Fig. 1. A requirements metamodel for WIS.

requirements are related to `quality scenarios`. More complex requirements classifications have been avoided, as the possibilities are countless, and greatly depend on the designer's preferences. Requirements which are organized into tree-like structures (see e.g. quality models such as the ISO 9126 [27]), along with relationships between functional and non-functional requirements, can be defined in a simple manner by means of the unary relationship `decomposedInto` which is defined over the requirement concept.

Each requirement has a set of relationships with other elements. First, the metamodel includes the `goal` concept, borrowed from the goal-oriented RE, as a means to model the high-level objectives of one or more stakeholders. Each goal, which can be decomposed into subgoals, is related to the `stakeholder` (one or more) that proposes it, and is satisfied through the fulfilment of a set of requirements.

Fig. 1 also contemplates requirements reuse by introducing the `catalogue` concept, which represents a set of related requirements. In a nutshell, a catalogue puts together a set of requirements extracted from one or more sources (i.e., a law, an organization policy, a particular domain, a guideline, etc.) and can be reused in all the projects in which these sources are applicable. This concept has been successfully applied in traditional RE methods [52,53] and its adaptation may be useful in the context of web-based projects, where numerous concepts, such as standard quality models [27], web accessibility guidelines and laws [55,22], experts usability recommendations [43], etc. must receive attention. These requirements sources have been formalized in our proposal by using the `source` element. A catalogue of requirements can be extracted from each source. The concepts of catalogue and source have certain advantages. On the one hand, they help the stakeholders involved in a web-based development project to discover the numerous guidelines, laws, recommendations, etc. involved in the development. These guidelines are not always used and, on many occasions, inexperienced practitioners do not know all of them. On the other hand, when a web-based development project is forced to comply with a law or a guideline, practitioners need only go to the adequate catalogue and find the requirements with which their project must comply.

With the use of this metamodel, the stakeholders involved in the elicitation of requirements will have a better knowledge of the concepts that must receive attention during this stage. Moreover, the use of the tool that supports the metamodel (explained in Sections 4 and 5) will assist them to gather requirements and concepts, such as reuse, will allow them to obtain benefits with regard to their experience in previous projects. In Section 5 (see Fig. 6), an example of use which illustrates how the metamodel can be instantiated and can help stakeholders to put emphasis on the elicitation of requirements for their systems is shown.

3. Expressing usability requirements on WIS models for early usability evaluation

Usability is a critical quality attribute for the success of interactive software systems such as WIS [30]. A great number of studies have highlighted the benefits of considering usability such as, for example, the improvement of users' productivity or the reduction in training and documentation costs. Furthermore, the investments made in usability have been shown to return economic benefits [11], which has motivated important organizations such as IBM or Boeing Co. to consider usability as a relevant factor in their software products.

However, the consideration of usability in the development of software faces certain obstacles. As it was previously mentioned in Section 2, one of these is related to the absence of the consideration of usability requirements in early phases of the development lifecycle. In addition, Seffah and Metzker [50] highlights other problems, which are that the activities related to usability are usually decoupled from the mainstream software development process, the use of notations and tools with which to consider usability are different to those used in the development process, or the lack of automatic support for some usability activities.

An analysis of the methodologies and tools traditionally used for WIS development reveals that these obstacles continue to be present in the scope of WE. For example, these methodologies are not concerned with the consideration of quality attributes such as usability during the development process. This task is usually delayed until the system has been completely developed, using tools called *usability and accessibility validators* (see [28] for detailed information about these methods) which validate the HTML and CSS code of the WIS. Thus, the possibility of moving some of these validations towards earlier phases in the development cycle, such as design, and their integration in the methodologies, are not considered. Another example of the problems mentioned by Seffah and Metzker [50] can be observed in the use of quality metrics within the scope of WIS. Ruiz et al. [48] carried out a survey of these metrics and discovered that up to 385 quality metrics had been defined with the aim of measuring quality attributes over this kind of systems. Half of the defined metrics are related to usability but, as Ruiz et al. [48] argues, they are not usually defined in a precise way and are not supported by any tool or, if this support exists, it is offered by external tools which are independent of those used in the WIS development process.

While in Section 2 we emphasized the gathering of requirements, such as usability requirements, in early phases of the development cycle as a means to obtain numerous benefits, the following sections show an approach with which to consider the definition and evaluation of usability requirements and metrics over the conceptual models used by WIS development methodologies. This approach arose after discovering that, during the elicitation of WIS requirements, stakeholders had proposed certain usability requirements than could not be expressed over the models used in WIS development. Thus, an extension of the navigational models that permits the expression of those requirements which cannot, at present, be expressed in design time has been defined. A study of the usability metrics that can be evaluated over WIS with the aim of improving its quality has also been carried out, along with the development of a prototype tool which permits the evaluation of the aforementioned usability requirements and metrics, and which can be integrated in the CAWE tools used by WIS development methodologies.

3.1. Usability on navigational models

Software usability is a quality attribute for which it is difficult to find a standard definition. The ISO 9126 standard [27] defines it as "the capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions". Nielsen [42] defines it as "the learnability and memorability of a software system, its efficiency of use, its ability to avoid and manage user errors and user satisfaction" and the ISO 9241-11 standard [26] as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use".

These multiple definitions imply that usability has been interpreted from different perspectives and that different stakeholders perceive it in different ways [50]. These definitions also show that usability is an abstract attribute which depends on numerous factors, meaning that if we wish to improve the usability of a WIS, we must pay attention to multiple features. The approach presented in this paper centres on usability as it is perceived by an end user, that is, as a quality attribute which allows end users to perform the expected tasks more efficiently. Although this definition is a little more precise than those previously mentioned, it continues to affect numerous features such as the intuitive navigation in the system, ease of use, simplicity in carrying out tasks or a comfortable and attractive presentation. Our approach is centred on improving the usability of a WIS by improving the navigation and the access to the information and functionalities that it presents, that is, by offering an intuitive navigation that makes it easy for the users for whom the WIS was conceived to carry out their tasks.

The approach was developed by studying the existing methodologies for WIS development and the models used by them to develop systems. Although each methodology presents specific features, in general, all of them use different models, such as navigational models (to model the interaction between users and the WIS), behaviour models (to model the data and functionality offered by the system) and presentation models (to represent features related to the final presentation of the system). By following a model driven approach, these models drive the development process and serve as a basis for the construction of the final WIS.

Our approach is centred on navigational models. Although the navigational models used by each methodology present slight differences [18], they are usually composed of two main elements: nodes and links between nodes. A node is used to represent a set of information or functionality that will be presented to WIS users. Links are used to join nodes, indicating the possibility of navigating from one node which represents a piece of information or a functionality to another. Other navigation structures such as menus or indexes may also appear in these models. Fig. 2 shows an example which may be useful in the observation of these concepts. It is a fragment of a navigational model for a WIS through which to purchase books via the Internet, and it presents information and functionalities related to their purchase.

The quality of navigational models is important since these models represent the possible paths that users may follow whilst navigating via WIS. Thus, errors in these models or less useful navigation designs influence the usability of the WIS which is finally developed. A number of studies centred on the evaluation and improvement of the quality of these models have been carried out. For example, Abrahão et al. [1] defines a set of metrics to provide modellers with information about the quality of navigational models, and Molina et al. [39] proposes a similar strategy for the verification and demonstration of properties of navigational models. Dhyani et al. [10] and Ruiz et al. [48] analyse and summarize the research carried out by numerous authors with regard to the definition of metrics to evaluate the quality of the different artefacts that participate in WIS development, but note that one of the main problems of these metrics is that they do not have automatic support or that it is offered by isolated tools which are not integrated in WIS development tools.

Our approach uses the positive features of the aforementioned proposals and additionally complements them with two improvements. Firstly, it not only allows modellers to use predefined metrics, but also permits them to use their own defined usability requirements and metrics over navigational models. Secondly, it offers an automatic support for these requirements which, when integrated in WIS development tools, may permit modellers a comfortable means through which to take advantage of the approach. The following subsection gives further details of how the first improvement can be carried out.

3.1.1. Evaluating usability requirements in modelling time

During the analysis of the requirements elicitation process for WIS carried out to develop the proposal presented in Section 2, we discovered that the stakeholders involved in the WIS development had proposed certain usability requirements which could not be expressed over the models used by WIS methodologies to build the system. The existence of this kind of requirements has been stressed in [3] and the application of the approach presented in Section 2 allowed us to discover some of them.

Some of these requirements were related to access to WIS information and functionalities. For example, some stakeholders wished to establish the maximum number of clicks that an end

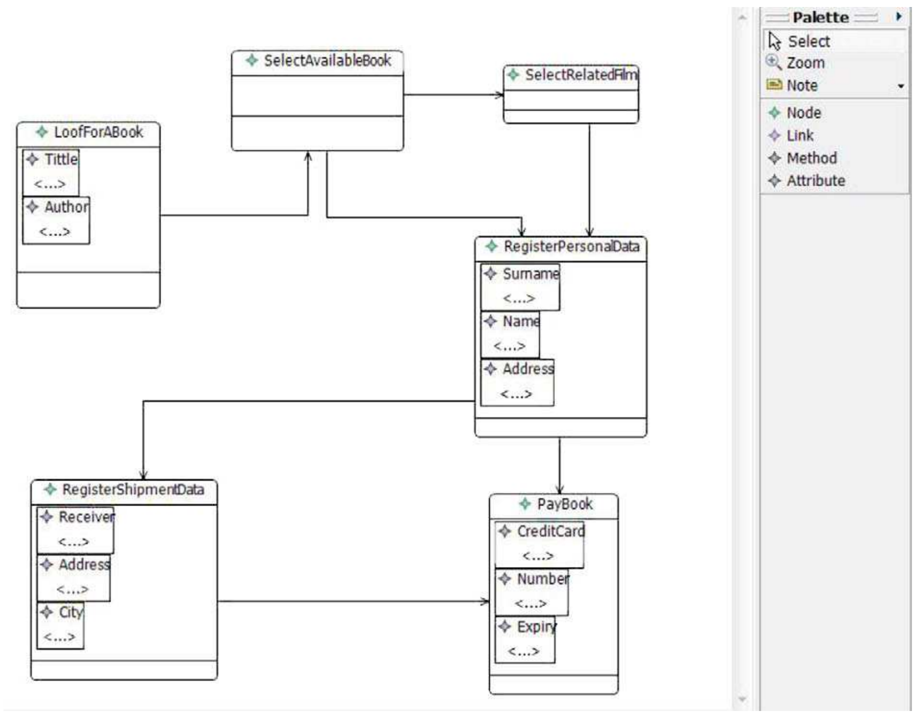


Fig. 2. An example of a navigational model.

user would need to make in order to carry out a concrete task. On other occasions, they wanted to express that not all the information and functionalities had the same importance, bearing in mind that primary information and functionalities related to the main aim of the system, and other information or functionalities that can be considered as secondary, usually exist in a WIS. For example, in the WIS through which to purchase books shown in Fig. 2, the primary functionality must be that related to searching for books and buying them. Other information or functionalities such as, for example, buying a DVD associated with the book may also be shown. This information is important but it is not the main aim of the system. In these cases, the stakeholders wanted this primary information to be close to the end users, for example, in terms of proximity to the entry point of the WIS.

On other occasions, the modellers wished to establish constraints relating to the order in which users visit nodes, or they wished to force the existence of direct or indirect connectivity among the nodes that represent related functionalities, under the premise that this facilitates navigation through the system, thus improving its usability. Table 1 summarizes some of these requirements.

Since this kind of requirements cannot be expressed during WIS modelling, they are usually forgotten and are not taken into account during the development process. This may mean that the WIS which is finally developed does not permit an intuitive navigation, and may lead users to feel lost and disoriented. In addition, the opportunity of obtaining the aforementioned benefits of usability analysis based on models is missed. To attempt to fill this gap, we have developed an extension of the navigational models which permits the expression and evaluation of this kind of requirements in modelling time. We will use the set of requirements shown in Table 1 to illustrate our approach, which is explained in the following subsection.

3.1.2. Extending navigational metamodels with usability features

The solution to the problem of allowing modellers to express the aforementioned usability requirements over their models consists of extending the metamodel of the navigational models with those entities, relationships and attributes that will allow this information to be collected. As was mentioned in Section 3.1, the two main elements of which navigational models are composed are nodes and links. We shall use these elements to illustrate our approach, considering a simplified fragment of the navigational metamodel (see top left of Fig. 3).

Support has been offered to the kind of requirements shown in Table 1, by extending the metamodel of navigational models in the following manner:

– The addition of new attributes. These attributes (called *MaxDistance* and *MinDistance*) are added to the element *NavigationNode* and are used to support the requirements related to distances between nodes.
– The addition of new links. Two recursive links have been added for navigation nodes (called *previousNodes* and *laterNodes*). These links represent, for each node, a list of previous and later nodes that a user must also visit if s/he visits that node and which are useful in supporting the kind of requirements related to navigation constraints.
– The addition of new elements. The *Level* entity and its attributes are used to support the requirements related to the importance of functionality and information. This entity represents the concept of *importance of a node* and its link with the *NavigationNode* entity allows modellers to label each node with an importance level. Each *Level* has three attributes: a name and the integers *MaxLevelDistance* and *MinLevelDistance*, which define the minimum and maximum distance between the nodes labelled with a *Level* and the node that represents the entry point to the WIS.

The extended metamodel obtained after adding these elements is shown at the bottom of Fig. 3. It is important to emphasize that the set of requirements and the extension of navigational metamodels presented are used to illustrate our proposal, but obviously other requirements which require new extensions can also be discovered and the approach could even be utilized on other models used in WIS development (e.g. presentation models) and not only on those which are navigational. In this case, we shall follow the same strategy, which consists of adding the attributes, entities and relationships that allow the expression of these new requirements over the models involved.

After analysing this extension and exploring its technical details in greater depth, we observed that various solutions could be used to obtain the extended metamodel:

(i) A direct extension of the navigational models proposed by WIS methodologies. In this case, the metamodel of the navigational models will be taken as the entry (see top left of Fig. 3), and all the new elements will be added to it. The result of this option is shown at the bottom of Fig. 3.
(ii) The creation of an independent metamodel that unites those elements (attributes, entities and relationships) which are necessary to support the new requirements (see top right of Fig. 3), which is then combined with the original navigational metamodel (see top left of Fig. 3) by using model transformations.

Table 1
A set of usability requirements that cannot be expressed in navigational models.

Goal	Requirements	Rationale
Definition of importance levels	Information/functionalities in the WIS must be organized into various levels	Not all the information/functionalities in the WIS have the same importance
Distances between nodes	To define Max/Min distance from the entry point to the WIS for each importance level	Possibility of detecting nodes labelled as important too far from the entry point and nodes labelled as less important excessively near
	To define Max/Min distance from the entry point to the WIS for each node	Possibility of detecting nodes which are little accessible for users
	To define distances between given nodes	If two nodes represent information or functionalities related to each other, modellers probably wish that they will be near
Connectivity constraints	To establish direct connectivity between nodes	It may be useful to express the requirement that two nodes must be directly connected
	To establish indirect connectivity between nodes	It may be useful to express that a node must be reachable from another one
Navigation constraints	To force previous crossing by a node	For each node it is possible to define a set of nodes that must be previously visited before reaching it
	To force later crossing by a node	For each node it is possible to define a set of nodes that must be visited after visiting it

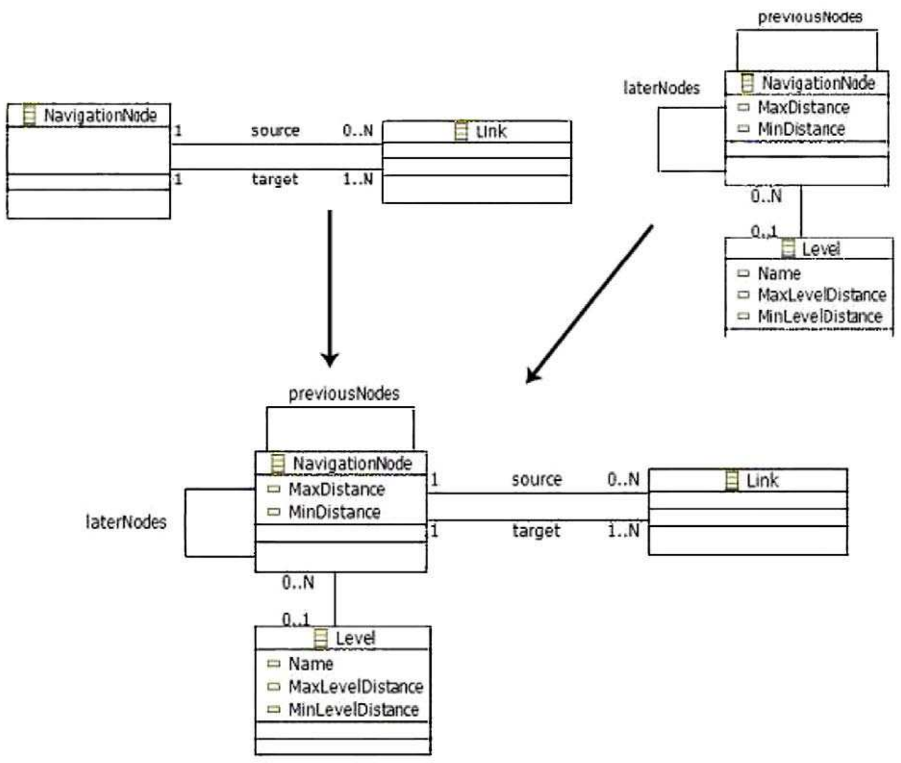


Fig. 3. Two options with which to extend navigational metamodels.

For our purposes, we have chosen the second option. The main problem of the first option is that it creates a hard coupling between the elements of which the navigational metamodels were originally composed and the new added elements. This might limit the extensibility of the proposal and it would be difficult for modellers to distinguish between the elements of the original metamodel and the elements added to support usability features. On the contrary, in the second option the original metamodels and the extensions used to consider usability features over them remain decoupled. It facilitates the possible extension of the approach because, to support new requirements, we only need to focus on the metamodel that combines the new elements. As exchange, it will be necessary to define the transformation which will obtain the target metamodel from the separate metamodels. This means of solving the merging of models and metamodels corresponds to a special kind of model transformations called model weaving [5]. Model weaving tackles the problem of given a model m_a which conforms to a metamodel M_a and a model m_b which confors to a metamodel M_b, then an integrated model m_{ab} can be obtained which conforms to a woven metamodel M_{ab}. As the reader may observe, this situation is similar to that presented above: we have two metamodels (the original navigational metamodel (M_a) and the metamodel created in order to consider usability features (M_b), which are shown at the top of Fig. 3) and we are interested in obtaining navigational models with usability features, which must conform the metamodel shown at the bottom of Fig. 3.

The weaving transformations can be specified in any transformation language such as, for example, QVT [44], the OMG standard for this aim within the scope of the MDE paradigm. Fig. 4 shows a fragment of the transformations that permit weaving between models m_a and m_b (that is, a navigational model and a usability features model which conform to their respective metamodels presented at the top part of Fig. 3) in order to obtain the model m_{ab}. These transformations have been defined by using the *QVT Relations* language, one of the languages defined in the QVT specification. Owing to lack of space, we can only show how the main entities of each metamodel are joined together in the woven metamodel (M_{ab}). These are:

– From M_b NavigationNode to M_{ab} NavigationNode (Fig. 4a). This QVT relation transfers the navigation nodes of a usability model ("nn" in Fig. 4a) to the target model, from now on called the weaving model ("wnn" in figure), using the attribute values of "nn".
– From M_b Level to M_{ab} Level (Fig. 4b). As in the previous relation the Level entities in the usability model are transferred into the weaving model.
– From M_a Link to M_{ab} Link (Fig. 4c). Finally, the Link entity of the navigational model is created in the weaving model. This relation selects a link in the navigational model and the references to its source and target node, and then matches the nodes in the weaving model that will be used to create the new link in the weaving model. This pattern matching is defined by using the variables "snnn" and "tnnn", which are bound to the adequate values in the models involved in the transformation. Moreover, we use the *when* clause in the relation to ensure that the two navigation nodes involved in the link have already been created in the weaving model.

Finally, it is important to highlight that the transformation presented here is not complete, since other features of the original model (such as associations, etc.) should also be transferred to the weaving model. However these other relations are not shown due to lack of space.

After the model weaving, it is time to evaluate the fulfilment of the usability requirements that have been expressed over the models built. With this aim, and using the OCL (Object Constraint Lan-

```
(a)
top relation QualityNavigationNodetoWeaving {
 nnn: String; maxD, minD: Integer;
 checkonly domain usability nn : NavigationNode {
   Name = nnn,
   MaxDistance = maxD, MinDistance = minD };
 enforce domain weavingMM wnn : NavigationNode {
   Name = nnn,
   MaxDistance = maxD, MinDistance = minD };
}

(b)
top relation LeveltoWeaving {
  ln: String; maxD, minD: Integer;
  checkonly domain usability l:Level {
     Name = ln, MaxLevelDistance = maxD,
     MinLevelDistance = minD  };
  enforce domain weavingMM lw:Level{
     Name = ln, MaxLevelDistance = maxD,
     MinLevelDistance = minD  };
}

(c)
relation LinkToWeaving {
 snnn, tnnn: String;
 checkonly domain navigational l : Link {
  source = snn:NavigationNode { name = snnn }
  target = tnn:NavigationNode { name = tnnn }
 };
 checkonly domain weavingMM
  swnn : NavigationNode { Name = snnn };
 checkonly domain weavingMM
  twnn : NavigationNode { Name = tnnn };
 enforce domain weavingMM l2 : Link {
  source = swnn, target = twnn };
when {
  QualityNavigationNodetoWeaving(snn,swnn);
  QualityNavigationNodetoWeaving(tnn,twnn);
 }
}
```

Fig. 4. A fragment of the model weaving transformations.

guage) plug-in [14] of the Eclipse platform, a set of OCL constraints has been implemented to guarantee that the usability features expressed are fulfilled. If any constraint is violated, modellers will be notified and will be able to modify their models in order to solve the detected problems.

3.1.3. Usability metrics for navigational models

Since the quality of navigational models influences the quality of the systems finally developed, the WIS research community has defined a number of quality metrics with the aim of improving the quality of these models. As our approach permits the definition and execution of OCL expressions over navigational models, an additional benefit of its application is that it can be used as an integrated framework for the definition and evaluation of usability metrics, which are usually defined in multiple isolated tools that are not integrated in the WIS development process. The WQM (Web Quality Metrics) model [48] was used to discover a set of evaluable metrics, and this carried out a review of literature in which it found up to 385 metrics defined to evaluate the quality of the artefacts used in WIS development. An analysis of these metrics has been carried out with the aim of selecting those that fulfil two requirements: (i) they must be related to the evaluation of WIS usability and (ii) it must be possible to evaluate them in modelling time, since our approach attempts to take advantage of the benefits of the model based usability analysis.

Among the metrics defined in WQM, 48% are related to usability and we have selected those which, since they are measurable in modelling time, are related to navigation through the system. Some of the usability metrics such as, for example, those that measure the number of broken links in the WIS finally developed, that count the number of graphics or are related to the navigation effort cannot be evaluated over models, since sufficient information for this purpose does not exist in modelling time. Nevertheless, a set of 50 quality metrics that can be defined and evaluated over navigational models already exist. The rationale behind these metrics is not within the scope of this work but more information is available in [48]. Lack of space prevents us from offering an exhaustive list of these metrics, but some examples are shown in Table 2. As we can see, these metrics offer quantitative information about the models (number of nodes and links, depth and width of the model, etc.), information concerning the connectivity among the elements in the model (relationship between the number of nodes and links, number of in and out links), etc.

After identifying these metrics, our next aim was to define and integrate them within the tool that supports the whole approach, as had been done with the usability requirements. This is shown in Section 5.2 as well as how the tool offers modellers the possibility of defining their own metrics and queries over their models by using OCL expressions.

3.2. Approach overview

Fig. 5 shows an overview of our approach using the SPEM notation [45]. It starts with the systematic elicitation of requirements using the proposed requirements metamodel and emphasizes the gathering of quality requirements such as those related to usability. After this phase, the requirements obtained must be expressed over the following models used to build the system. In our case, we focus on navigational models, which are built together with the usability model that offers modellers the capacity to express usability requirements which cannot at present be defined over these models. In the weaving phase, both models are combined to obtain navigational models extended with usability features. After the system has been designed, the fulfilment of the usability requirements is checked by means of the execution of the OCL constraints which are predefined with this aim. Moreover, in this phase modellers execute the implemented usability metrics over their models in order to obtain information about their quality. The information obtained after checking the requirements and evaluating the metrics is used by modellers to, if necessary, improve the models in their system. Finally, the models obtained will be used to implement the system.

Table 2
Some examples of usability metrics for navigational models.

Goal	Examples of metrics
Quantitative information	Depth, width, diameter, radius, number of nodes, number of links, number of attributes and methods, etc.
Model connectivity	Number of in and out links, compactness, stratum, average connected distance, connectivity density, etc.

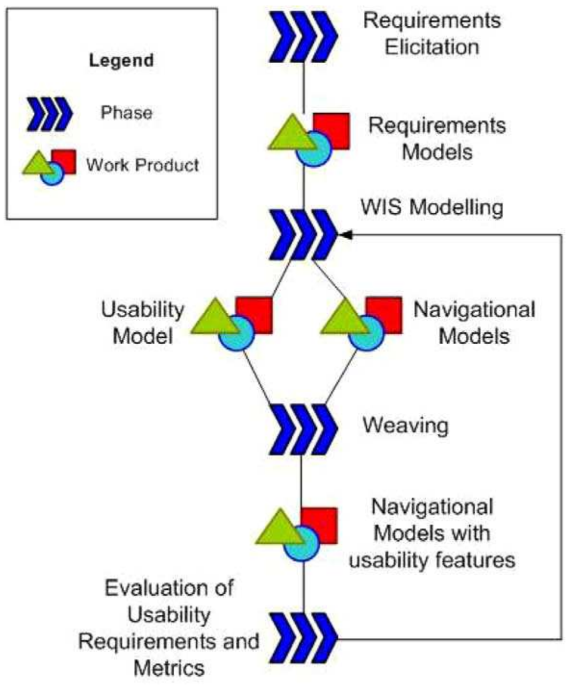


Fig. 5. Approach overview.

4. Automatic support

As was previously mentioned, one of the key factors for the successful handling of usability in the WIS development process is the existence of an automatic support for the approach which can be simultaneously integrated in the tools usually used in the WIS development. We have, therefore, designed a tool which supports the presented contributions and allows stakeholders to manage the concepts involved in a comfortable manner. The following sections explain the considerations taken into account when choosing the appropriate technological space in which to develop this tool.

4.1. Technological environment

Certain considerations had to be taken into account when selecting the technological environment in which this tool would be implanted. First, it was necessary to use an environment that would permitted the easy definition of metamodels, since our approach involves both a requirements metamodel and the extensions carried out to consider usability on navigational models. Furthermore, as we wished various stakeholders (and not only designers) to use our approach, we needed to offer them a graphical tool with a usable and comfortable interface which would allow them to create and manipulate models compliant with our approach and which would also facilitate the evaluation of the quality of their models. Furthermore, the presented approach was not thought to be an isolated effort but an effort oriented towards its integration in WIS development processes with the aim of reinforcing both the requirements management and usability evaluation activities. Therefore, the capacities offered by the technological environment in order to extend the tool with new functionalities or to integrate it in existing tools used in WIS development had to be considered.

Given these premises, the Eclipse platform and, in particular, the Eclipse Modelling Framework (EMF [12]) was selected. Eclipse and EMF offer certain suitable features that make them interesting for our approach. First, EMF offers support to deal with MOF, the standard that OMG recommends for describing metamodels. EMF is therefore useful for the creation and manipulation of models and metamodels such as those proposed in our approach. Second, it is an open source platform-independent project and the architecture, which is based on plug-ins, makes it easy to reuse and to add functionality. Moreover, Eclipse offers an OCL implementation which is useful in defining requirements and metrics in this language. Finally, a further important advantage of Eclipse is that some CAWE tools used for WIS development such as WebRatio [2] are being migrated to this platform and some of the efforts [38] to reach an agreement with regard to the concepts managed in WE use Eclipse as technological environment. The integration of our automatic support in these Eclipse tools through a new plug-in can be directly obtained.

The steps carried out for the implementation of this automatic support have been the following:

– Definition of the metamodels that allow both the requirements modelling and the design of navigational models with usability features (see Figs. 1 and 3), using the EMF project.
– Definition of graphical editors that allow stakeholders to define requirements models and navigational models with usability features in an easy and intuitive manner, by using a palette of elements similar to those included in any CASE tool. The models that can be defined with these editors conform to the metamodels defined in the previous step. The GMF (Graphical Modelling Framework [13]) project has been used for this purpose.
– Definition of the weaving transformation explained in Section 3.1.2 using Medini QVT [51].
– Definition and implementation of the checks that support the evaluation of usability requirements and metrics. This task is supported through the implementation of OCL offered by Eclipse [14].

The following section will show the appearance and functionality of this tool through an example of its use.

5. Case study

This section illustrates how the presented approach can be used for the elicitation of requirements and the subsequent evaluation of usability attributes in design time through an example of its use which corresponds to a simplified on-line book sale system. The example has been developed using the automatic support implemented to support our approach.

For the sake of simplicity, let us assume that the CEO of a certain company has decided to expand her/his business strategy, and that her/his goal is to *attract new clients*. The CEO believes that offering the books through Internet may positively influence both the attainment of new clients and the reduction of the cost associated with the sales process, so s/he has embarked on a web book sales system development process. Both the CEO and the web customer who will interact with the application have as their main functional requirement *buy books*. This requirement can be decomposed into two functional requirements: *browse available books* and *purchase books*.

In addition, several non-functional requirements that the web based system must fulfil have been identified. Firstly, the buy books functionality should follow *accessibility* guidelines to allow web customers with disabilities to access the system according to the related laws and guidelines created for this aim such as Section 508 [22] or WAI recommendations [55]. Additionally, and in relation to usability attributes, the system should provide *information accuracy* while browsing through the available books: synopsis, prices and so forth should be reliable. Also, the application *learnability* should be high, that is, the application should be simple enough for novice users to easily learn its operation. With regard to navigation through the WIS, some requirements have been established. The CEO wants to offer, if they exist, the possibility of buying the films associated with each book, but s/he considers this information as secondary, so it must not be directly accessible from the entry point. The browse books functionality is the most important, so it must be at a maximum distance of one click from the entry point. In addition, it must not be possible to buy books if the user has not previously registered in the system and the functionalities to select a book and pay for it must be directly attainable. Finally, the purchase process should be performed assuring the *security* of the customer data.

5.1. Instantiating the requirements metamodel

If we check the metaclasses of the metamodel in Fig. 1, we can observe how:

(i) The CEO and web customer both instantiate the *Stakeholder* metaclass.
(ii) Attract new clients instantiates the *Goal* metaclass.
(iii) Buy books, browse available books or purchase books are all *Functional Requirement* occurrences.
(iv) The requirements related to accessibility, ease of navigation, learnability, security and information accuracy are all instances of *Non-Functional Requirement* instances.
(v) The WAI guidelines are a requirements *source*. If this source has been applied to other projects, the associated requirements can be directly incorporated into this system by using the corresponding requirements *catalogue*.

Logically, these requirements are decomposed into other more concrete which have not been shown for the sake of simplicity. The requirements metamodel significantly helps in the systematic identification of these requirements which in its turn serves as a good baseline to start the WIS modelling. These identified requirements must then be reflected on WIS models, such as content, navigation or presentation models.

Fig. 6 shows the automatic support for the requirements metamodel. After formalizing it by using EMF, a graphical editor which permits the construction of requirements models that conforms to the metamodel has been developed. On the right side, the tool shows a palette with which to manage the concepts involved (stakeholders, goals, requirements, etc.), and this is next to an intuitive icon which is used to build the requirements models. In the figure, only a fragment of the application example is shown, but the complete model would of course include more requirements, along with other stakeholders, sources, etc. Until now, the goal proposed for the CEO of the company has been expressed together with some of the requirements necessary to fulfil them. The `Navigation constraints` requirement must be divided into the other requirements related to navigation mentioned in the description of the application example. Since the WAI recommendations are used in multiple projects, it is possible to take advantage of the concept of reuse thanks to the requirement catalogue presented in the model.

5.2. Designing navigational models with usability features

After the requirements elicitation phase, the following step consists of building the conceptual models that express the gathered requirements. Further details of how the requirements related to ease of navigation can be represented on our navigational models extension and how the quality of these models can be evaluated using the usability metrics defined over them will now be given.

As occurred with the requirements metamodel, EMF and GMF have been used to define the metamodels for the navigational models and also to build the graphical editors that allow modellers to define these models comfortably. Fig. 2 shows a fragment of the navigation model for our example application, which has been modelled using the graphical editor.

Since these models have been extended to support usability features, modellers can express the requirements related to the navigation while they simultaneously build their navigational models. Fig. 7 shows how some requirements constraints can be defined in the usability features model. As the CEO established, there is information concerning two levels of importance, defined as `High` and `Low`, in Fig. 7. The nodes related to the sales of books have been labelled with the `High` Level and those related to the sales of films with the `Low` Level. Moreover, certain maximum/minimum distances have been defined for these nodes. As we can observe at

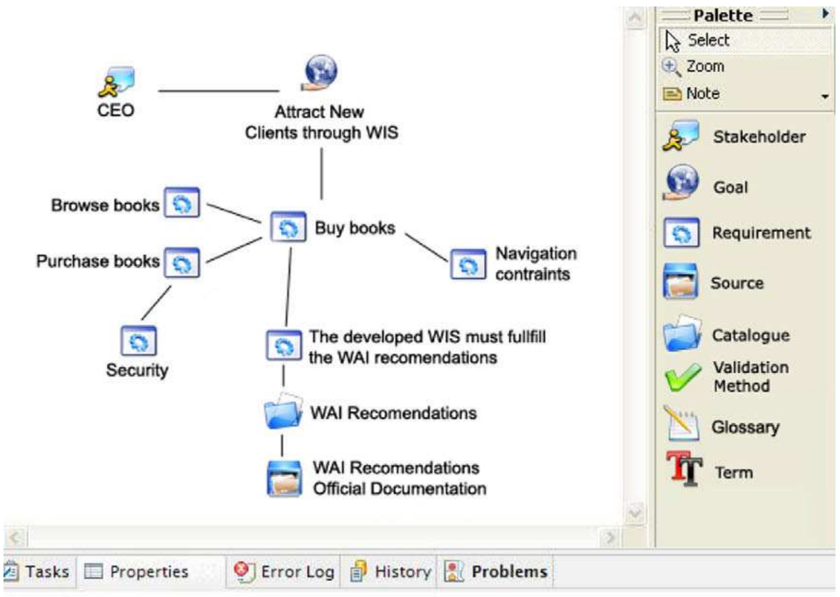
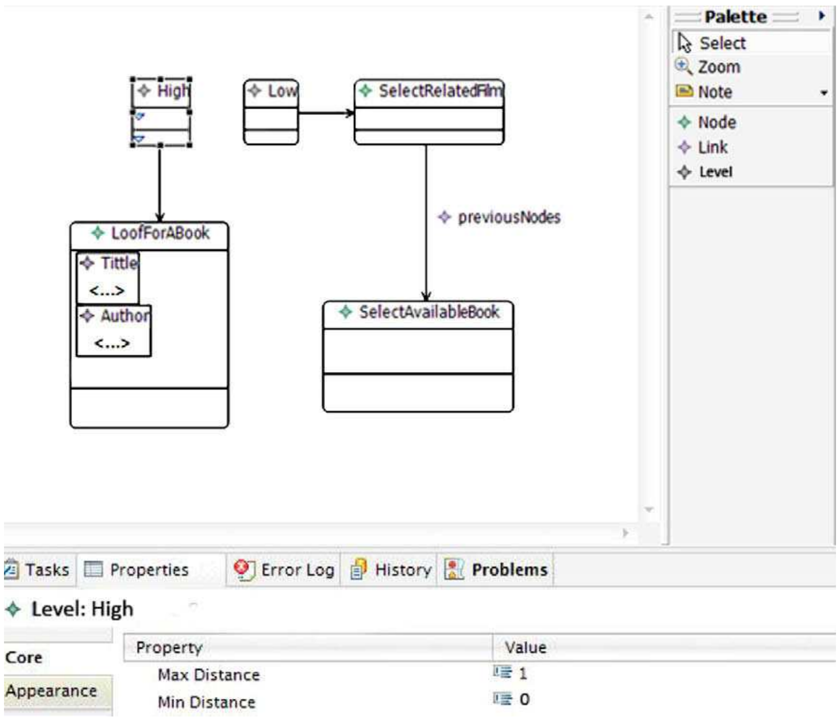


Fig. 6. A fragment of the requirements model for the case study.



Level: High

Property	Value
Max Distance	1
Min Distance	0

Fig. 7. Adding usability constraints to the navigational model.

the bottom of Fig. 7, the modeller has defined a maximum distance of one click among the nodes labelled with `High` and the entry point.

After their expression of these usability requirements, modellers can execute the proofs implemented to assure that these requirements are fulfilled. Furthermore, modellers also have at their disposal the implementation of the set of usability metrics mentioned in Section 3.1.3. The information obtained after execution provides them with valuable information with which to evaluate the quality of their models, and to improve them if necessary. Furthermore, the OCL plug-in allows modellers to define their own metrics over their models. Fig. 8 shows some straightforward examples of these metrics.

Fig. 8a shows an example of the kind of usability metrics shown in Table 1. In this case, it counts the number of nodes in the model. Fig. 8b is useful to illustrate one of the metrics that the modeller can define and execute over their models. In this case, the OCL constraint is defined over the model of Fig. 7 and selects those nodes labelled with a `High` importance level. Finally, it is important to emphasize that modellers can define not only metrics but even other kind of constraints over their models. For example, Fig. 8c illustrates a constraint that establishes that all the links in the model must have different names. These metrics and constraints can be incorporated in the tool and the modellers just select them and obtain information that can use to improve their models.

6. Conclusions and further work

The improvement of the quality of WIS must start from the early stages of the development lifecycle. Our approach makes it possible to deal with usability attributes from the requirements elicitation and early WIS design phases. It provides a requirements metamodel which helps in the elicitation of WIS requirements and complements the other models used by WE methodologies, which, until now, often began their development process from WIS design. After using this metamodel, more attention was paid to requirements elicitation, and a set of usability requirements which can be evaluated in design time has been discovered. These requirements have served as a basis to develop an extensible approach that allows modellers to define and evaluate usability requirements in modelling time.

Moreover, after studying the relevant literature, a set of usability metrics which can be evaluated over models have been identified and integrated in our approach, which now offers an integrated framework for these metrics. These help modellers to improve the quality of their models and avoid the necessity of using multiple isolated tools which only implement subgroups of these metrics.

As further work, and with regard to the reinforcement of the requirements management activities in WIS methodologies, we are conscious that our metamodel is a first step towards establishing the basis for new improvements. We are therefore refining it, and simultaneously developing a profile of the i^* notation as a means to reduce the complexity of object models when the size of the WIS projects grows. In addition, the connection of this metamodel with validation methods that help to validate the fulfilment of the collected requirements and the definition of transformation rules from requirements to other artefacts that participate in WIS development, such as navigational or presentation models, is being carried out. With regard to the activities of early usability evaluation, the approach can be extended in order to discover and incorporate more requirements and metrics which can be evaluated in design time, thus enriching the navigational models' quality extension. Moreover, as was mentioned in Section 3, a similar approach can be carried out on other models used in WIS development, such as presentation models. A set of experiments to help provide a more empirical evaluation of the proposal is also being defined.

```
(a) self.nodes->size()
(b) self.nodes->select(n:Node | n.level.Name = "High")
(c) self.links->isUnique(l: link | l.name)
```

Fig. 8. Some examples of OCL metrics and constraints.

References

[1] Abrahão S, Condori N, Olsina L, Pastor O. Defining and validating metrics for navigational models. In: METRICS 2003, Sydney, Australia. IEEE Press; 2003. p. 200–10.
[2] Acerbis R, Bongio A, Brambilla M, Butti S. Webratio 5: an eclipse-based case tool for engineering web applications. In: ICWE; 2007. p. 501–5.
[3] Atterer R, Schmidt A. Extending web engineering models and tools for an automatic usability validation. J Web Eng 2006;1:43–64.
[4] Berre AJ. Comet (component and model based development methodology); 2006. <http://modelbased.net/comet/>.
[5] Bézivin J. On the unification power of models. Softw Syst Model 2005;4(2): 171–88.
[6] Blaine JD, Cleland-Huang J. Software quality requirements: how to balance competing priorities. IEEE Softw 2008;25(2):22–4.
[7] Bolchini D, Paolini P. Goal-driven requirements analysis for hypermedia-intensive web applications. Requirements Eng J 2004;9(2):85–103.
[8] Ceri S, Fraternali P, Bongio A, Brambilla M, Comai S, Matera M. Designing data-intensive web applications. San Francisco, CA, USA: Morgan Kaufmann Inc.; 2002.
[9] Damian D, Chisan J. An empirical study of the complex relationships between requirements engineering processes and other processes that lead to payoffs in productivity quality and risk management. IEEE Trans Softw Eng 2006;32(7): 433–53.
[10] Dhyani D, Keong W, Bhowmick S. A survey of web metrics. ACM Comput Surv 2002;34(4):469–503.
[11] Donahue GM. Usability and the bottom line. IEEE Softw 2001;18(1).
[12] Eclipse. Eclipse modeling framework project (EMF); 2007. <http://www.eclipse.org/modeling/emf/>.
[13] Eclipse. Eclipse graphical modeling framework; 2008. <http://www.eclipse.org/gmf/>.
[14] Eclipse-OCL. Model development tools (MDT); 2008. <http://www.eclipse.org/modeling/mdt/?project=ocl>.
[15] Epner M. Poor project management number-one problem of outsourced e-projects. Research Briefs, Cutter Consortium; 2000.
[16] Escalona M^a J, Aragón G. NDT a model-driven approach for web requirements. IEEE Trans Softw Eng 2008;34(3):377–90.
[17] Escalona M^a J, Koch N. Requirements engineering for web applications: a comparative study. J Web Eng 2004;3:193–212.
[18] Escalona M^a J, Torres J, Risoto M, Gutiérrez JJ, Villadiego D. The treatment of navigation in web engineering. Adv Eng Softw 2007;38(4):267–82.
[19] Ginige A, Murugesan S. Guest editors' introduction: web engineering – an introduction. IEEE MultiMedia 2001;8(1):14–8.
[20] Goknil A, Kurtev I, van den Berg K. A metamodeling approach for reasoning about requirements. In: ECMDA-FA; 2008. p. 310–25.
[21] Gómez J, Cachero C, Pastor O. Conceptual modeling of device-independent web applications: towards a web engineering approach. IEEE Multimedia 2001;8:26–39.
[22] USA Government. Section 508: the road to accessibility; 2007. <http://www.section508.gov/>.
[23] Greenfield J, Short K, Cook S, Kent S. Software factories: assembling applications with patterns models frameworks and tools. John Wiley & Sons; 2004.
[24] Standish Group. The chaos report; 2002. <http://www.standishgroup.com/>.
[25] Insfrán E, Fernandez A. A systematic review of usability evaluation in web development. In: WISE workshops. LNCS, vol. 5176; 2008. p. 81–91.
[26] I.S.O. Iso 9241-11. Ergonomic requirements for office work with visual display terminals (vdt)s – part 11: guidance on usability; 1998.
[27] I.S.O. Iso/iec 9126-1. Softw. eng.-product quality – part 1: quality model; 2000.
[28] Ivory MY, Hearts M. An empirical foundation for automated web interface evaluation. PhD thesis, University of California at Berkeley; 2001.
[29] Juristo N, Moreno AM, Isabel Sánchez M. Analysing the impact of usability on software design. J Syst Softw 2007;80(9):1506–16.
[30] Juristo N, Moreno AM, Sanchez-Segura M^a I. Guidelines for eliciting usability functionalities. IEEE Trans Softw Eng 2007;33(11):744–58.
[31] Kappel G, Michlmayr E, Pröll B, Reich S, Retschitzegger W. Web engineering – old wine in new bottles? In: ICWE; 2004. p. 6–12.
[32] Kleppe AG, Warmer J, Bast W. MDA explained: the model driven architecture: practice and promise. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.; 2003.
[33] Koch N, Kraus A. The expressive power of uml-based web engineering. In: Second international workshop on web-oriented software technology (IWWOST); 2002.
[34] Koch N, Zhang G, Escalona M^a J. Model transformations from requirements to web system design. In: ICWE; 2006. p. 281–8.
[35] Lang M, Fitzgerald B. Web-based systems design: a study of contemporary practices and an explanatory framework based on method-in-action. Requirements Eng J 2007;12(4):203–20.
[36] Lowe D. Web system requirements: an overview. Requirements Eng J 2003;8(2):102–13.
[37] Matera M, Rizzo F, Cortázar R, Perallos A. The usability dimension in the development of web applications. In: Handbook of research on web information systems quality. Idea Group Publishing; 2008. p. 235–49.
[38] MDWEnet-Initiative. Web engineering metamodel; 2008. <http://www.pst.ifi.lmu.de/projekte/mdwenet/index>.
[39] Molina F, Lucas FJ, Toval A, Vara JM, Cáceres P, Marcos E. Towards quality web information systems through precise model driven development. In: Handbook of research on web information systems quality. Idea Group Publishing; 2007. p. 317–55.
[40] Molina F, Pardillo J, Toval A. Modelling web-based systems requirements using WRM. In: II International workshop on web usability and accessibility. WISE workshops. LNCS, vol. 5176; 2008. p. 122–31.
[41] Moreno N, Romero JR, Vallecillo A. An overview of model-driven web engineering and the MDA. In: Web engineering and web applications design methods; 2007 [chapter 12].
[42] Nielsen J. Usability engineering. Boston, MA, USA: AP Professional; 1993.
[43] Nielsen J. Designing web usability: the practice of simplicity. New Riders Publishing; 2000.
[44] OMG. MOF QVT standard specification; 2005. <http://www.omg.org/docs/ptc/05-11-01.pdf>.
[45] OMG. Object management group. Software process engineering meta-model V1.1; 2005. <http://www.omg.org>.
[46] Rashid A, Moreira A, Tekinerdogan B. Special issue on early aspects: aspect-oriented requirements engineering and architecture design. IEE Proc – Softw 2004;151(4):153–6.
[47] Robertson S, Robertson J. Mastering the requirement process. Boston, MA, USA: Addison-Wesley; 1999.
[48] Ruiz J, Calero C, Piattini M. Classifying web metrics. In: Software audit and metrics; 2004. p. 22–37.
[49] Schmidt DC. Guest editor's introduction: model-driven engineering. IEEE Comput 2006;39(2):25–31.
[50] Seffah A, Metzker E. The obstacles and myths of usability and software engineering. Commun ACM 2004;47(12):71–6.
[51] Ikv++ Technologies. Medini QVT 1.4 ; 2008. <http://www.ikv.de/>.
[52] Toval A, Moros B, Nicolás J, Lasheras J. Eight key issues for an effective reuse-based requirements process. J CSSE 2008;23(5).
[53] Toval A, Nicolás J, Moros B, García F. Requirements reuse for improving information system security: a practicioner's approach. Requirements Eng J 2002;6(4):205–19.
[54] Vicente-Chicote C, Moros B, Toval A. Remm-studio: an integrated model-driven environment for requirements specification, validation and formatting. J Object Technol 2007;6(9):437–54 [special issue TOOLS EUROPE 2007].
[55] World Wide Web Committee (W3C). Web accessibility initiative (WAI). <http://www.w3.org/wai/>.
[56] Yamamoto S, Kaiya H, Cox K, Bleistein SJ. Goal oriented requirements engineering: trends and issues. IEICE Trans 2006;89-D(11):2701–11.
[57] Yu ESK. Towards modelling and reasoning support for early-phase requirements engineering. In: RE, IEEE Computer Society; 1997. p. 226–35.