The Modelery: A Model-Based Software Development Repository

R. Couto, A.N. Ribeiro, and J.C. Campos

1. Introduction

Research into software development processes typically produces large amounts of artifacts, from documentation and different kinds of models to the actual code. Organizing and sharing those artifacts has shown to be somehow a difficult task, due to the lack of effective support. We are particularly interested in the development of tools and techniques to support software engineering and reengineering (c.f. Couto et al., 2012, Campos et al., 2012, Campos and Harrison, 2009), and the problems faced by teams applying them. The amount of produced artifacts when using these tools, and (in many cases) the distributed nature of the teams, begs the question of how to adequately store, catalog, archive and share such artifacts. It becomes all too easy to lose track of existing versions, the relations between artifacts, and even the artifacts themselves.

The use of standard version control systems (such as Subversion (SVN)) has shown to be inadequate (France et al., 2006a). In fact, it is not our objective to have a system with version control capabilities, as delta updates. Instead, we aim towards a repository for a diversity of artifacts. By artifacts, we are referring to the inputs and outputs of a software (re)engineering process, but mostly models. Examples of artifacts include different types of models, test cases, pattern catalogs, processes descriptions, software prototypes, meta-models, or database schemes. Despite being a repository able to store generic software engineering artifacts, we will mainly refer to models in this paper.

Three main functionalities are considered relevant in this context: repository functionalities (archive, catalog, categorize, search, explore and share capabilities); social functionalities (groups support, associating groups with artifacts); scientific publications support (management and association with scientific publications). We classify such platform as a collaborative Web repository. On the one hand, it allows multiple researchers to collaborate in a project through a Web environment. On the other hand, it provides archiving capabilities (i.e., a repository). We consider a Web information system to be the best solution to access this type of system. It ensures that the users will be able to access it from almost any device with a Web browser, without the need to install any software. Some Web 2.0 functionalities, such as

dynamic content and user supported contents (i.e., forums), improve both the interaction of the users with the platform, and among them.

In this paper we present and discuss the implementation of the Modelery, a platform aimed at providing the functionalities just discussed. A previous version of the platform was described by Couto et al., 2014a. This paper extends that work by reviewing the related work introducing new platforms, and presenting the improvements on the Modelery over the last version. Improvements include a new presentation framework (Java Server Faces), which lead to the reimplementation on part of the tool and implementation of the major functionalities as web services. The reimplementation lead also to simplification and refination on some functionalities, as for instance the artifacts search.

The paper is structured as follows: Section 2 reviews related work, with the analysis of a number of similar tools; Section 3 builds on that to present the requirements for the platform; in Section 4 the tool is described; we then present an applicability study of the framework in Section 5; finally, Section 6 presents some discussion about what has been achieved, and Section 7 concludes the paper with some pointers for further work.

2. Related work

Current collaborative repository tools can be categorized into two main approaches: data repositories and process model repositories. In this section we analyze existing tools in each category, evaluating how suitable for our purposes they are. This analysis provides also valuable input regarding the requirements for this type of platform. Table 2 (see Section 6) presents the comparison of the discussed platforms regarding their functionalities. This section presents the most relevant tools.

2.1. Data Repositories

Data repositories are common among the database research communities. They are the extension of a database management system, with emphasis on metadata management. The repository consists in a “shared database of information about engineered artifacts produced and used by an enterprise” (Bernstein and Dayal, 1994). Model management systems are also related with data repositories, addressing problems of models representation and processing (Dolk and Konsynski, 1984).

ReMoDD (Repository for Model Driven Development) is a Web platform developed by the Colorado State University Department of Computer Science and Engineering (France et al., 2006b). This platform aims to support ways to share models, information, case studies and knowledge among multiple audiences, for instance teachers, researchers and students. The platform provides the basic repository functionalities. It has artifacts listing and browsing, sorted by several criteria: name, description, categories, author(s) or update data. There are no further artifacts' discovery functionalities. By opening a model it is possible to visualize its details, post comments and download it. The artifacts details provide only general information, and lack for instance the authoring tool, scope and version. The groups functionality is also not deeply integrated. Finally, at the moment, the platform is not accepting registrations.

The Software-artifact Infrastructure Repository (SRI), from the University of Nebraska, is another repository in this case specifically for software artifacts. It is meant for "supporting rigorous controlled experimentation with program analysis" (Do et al., 2005). It contains Java, C, C++ and C# software systems. It supports storing and searching artifacts, as well as showing details and downloads. This repository is not directly suitable for our needs, as it allows only four types of documents. Also, the search and browsing functionalities are somehow limited.

A number of other platforms include some form of repository but are more focussed in supporting specific aspects of a software engineering process. The ATL Transformations zoo, is a static repository of ATL (a model transformation language (Jouaulta et al., 2006)) transformation programs, presented in the form of a list of artifacts. It is accessible both in a Web page and via the Eclipse IDE. Despite its static nature, the integration with the IDE is a feature that we found interesting and worth exploring. GenMyModel (Dirix et al., 2013) provides model editing and storing functionalities. This commercial platform is more focused in model editing than in the repository functionalities. Colex is a model repository that focusses in model versioning and conflict resolution (Brosch et al., 2010). This repository targets specifically models expressible in XMI (OMG 2014).

Software Engineering is not the only area where repositories have been used. ECOBAS is a Web information system designed for ecology and environmental sciences (Cavalcanti et al., 2002). It supports online modeling and simulation, but also offers some interesting repository functionalities. It provides both a Web interface and local client. The Web interface allows users to search a model by name,

by subject or by free-text. Viewing the models' information is similar to other repositories. It is possible to select a model from a list, and its details are presented. The focus of this platform on ecological and environmental context makes it unsuitable for our purposes. However, analyzing the tool made us aware of the importance of having an open platform. A flexible platform should provide support for a large variety of models, regardless of their application area. Another limitation of ECOBAS is the information shown about each model, which despite being detailed misses some relevant information such as a visual representation.

We consider model organization, storing and categorization as the core of a model repository. Such functionalities are also found in books managing systems, as is the case of Shelfari. This platform provides a digital library to store and organize books. Book entries can be searched, listed, added, removed and rated. Cataloging is done through several aspects, such as subject, author and tags. The concept of group is also present, where a set of users sharing the same interests about a particular subject can discuss it. While not directly usable for our needs, the tool provides useful hints for developing a new platform, as the task of cataloging artifacts shares some concepts with cataloging books.

2.2. Business Process Model Repositories

Business process model repositories, are based in workflow and conceptual modeling. They provide a repository and execution environment for those models (Rosa et al., 2011).

Apromore (Rosa et al., 2011) is one example of Business Process Model repository. While it was possible to test a first version of the repository, that version has since then been deprecated and taken offline. A new version of the tool is under development but is currently unavailable to test. Hence the current analysis refers to the deprecated version. Apromore provides model storage and management functionalities (both view and create/edit). The models' discovery functionalities are adequate, as they support listing, searching and filtering of models (by criteria). All the models' details are available, and it supports rating the models. This tool provides an intuitive user interface for model management. However, groups are not supported, and all models exist at the same level, being available to all users (there is no visibility concept). This platform is closer to a repository than to a collaborative environment. Additionally, it supports only the storage of models created directly in the platform.

A number of commercial platforms exist which provide some level of repository functionality, although that is not their main focus. Examples include ARIS, an

enterprise architecture management with an emphasis in business process models; Adonis (Karagiannis and Kühn, 2002), which is focused in business process management; and ModeleR (Pérez et al., 2012), an example from the environmental management and ecological research domains. One of the features of this latter system is its support for model execution. We are not considering server side model execution at this stage.

2.3. Discussion

None of the analyzed tools was found suitable to fulfill the needs of a collaborative Web repository which can fully support the archiving, sharing and dissemination of models or other software engineering artefacts. Briefly, it is possible to say that the tools are either for a specific domain, for a specific language, are closed (for registration), or are too limited in functionalities. A platform that seems promising is ReMoDD. However, a set of limitations (not the least of which is the fact that it is currently not accepting further registrations) makes it inadequate for our objectives. Additionally, the platform lacks Web 2.0 functionalities to encourage collaboration between researchers (Brosch et al., 2010).

3. Requirements for a collaborative Web repository

Combining the functionalities we had initially identified with the information extracted from the analysis above, allowed us to define a set of requirements to guide the development of a new community supported (i.e., the models are provided) Web repository. This section presents these requirements.

To start with, the platform will require what Rosa et al., 2011 designate as the standard repository functionalities, which include data storage, access control, and simple search queries. Those requirements are not enough when developing a new system, if we want it to be better than existing solutions. Hence, it was decided that the new platform should include some other functionalities, such as advanced search functionalities.

3.1. Artifacts repository

The main functionalities that we look forward in a repository are model archiving and cataloging. Archiving models in a centralized platform will help keep track of their location, and their sharing with others. Cataloging the models allows storing them in a meaningful manner, and eases the process of finding them at a later stage. Cataloging enables also the possibility of other people finding the models. While a

user account is required to upload and manage models., read access to the repository is open to all.

Searching models by text is the most direct approach to perform searches. It is the norm in repositories and search engines in general. Textual search should support finding models through either their name or description. This approach will increase the probability of finding models within the repository.

Models are prone to changes and updates, and such factor is essential when developing a repository. In order to support such behavior we propose supporting several versions of the same model.

The decision of making a model public (accessible to everyone) or private is left to the user. Hence, the user might decide to keep a model private, for instance while in development, or only available to a subset of users. If a model is public, it should be accessible by anyone. If a model is private, only the author should be able to see and modify it. Lastly, in order to support collaboration, it must be possible to restricted a model's access to a group.

Users with access to a model should also be allowed to add comments and ratings, as well as being notified when new versions are deposited in the platform. This is where the collaborative functionalities start, in the sense that different users may cooperate in the development or improvement of a model.

We consider interoperability between applications to be essential. While using the Web page to interact with the repository might be the easiest way for human interactions, the same is not true for applications communications. Also, the interoperability allows to further extend the platform and to allow alternative methods to access the models (as is the example with ATL zoo). In order to support the interoperability we propose the implementation of a set of web services to perform the most common operations in the repository.

3.2. Publications management

The main approach to disseminate scientific results is through scientific publications. As space is typically limited it becomes useful to be able to point to outside sources for models and other artefacts resulting from the research. In this context it makes sense to manage references to scientific publications inside the platform, supporting their association with available models. As a model might also be referred in several articles, we propose a bidirectional relationship between models and publications. With this functionality it should then be possible to

reference or search for models related with specific publications, or conversely, search publications related with specific models.

3.3. Collaborative functionalities

It is common for the research process to involve interaction among several persons and ideas as well as with previous works. It is also well known that collaboration and sharing of information improves research results. From multiple people, different approaches emerge and sometimes best results are found by combining several peoples' ideas. This is the basis of collaborative platforms (Wang et al., 2010) .

One solution to support collaboration in the repository would be to integrate social functionalities with the models. The concept of groups of users, allied with forum functionalities, seems an appropriate requirement. By creating groups where the users can discuss ideas, and associating models to them, we aim to foster a collaborative behaviour amongst the users of the platform.

In the same way that models have a visibility option, it makes sense to have the same option for groups. Hence, it should be possible to make a group (as well as its models) restrict to a set of users. With this approach only the subset of persons related with the project will have access to the group's information. This is especially useful for private projects, or projects still under development. When a model is part of a group, it would be adequate to allow both the author and the members of the group to update it.

3.4. Levels of sharing

Not all models and groups are developed for the same purpose. Some of them are intended to be public, other restricted to a subset of persons (and updateable by all these persons, or by the author only) and other completely private. Also the groups may themselves either be public or private. The distinction between all these visibility levels is crucial to cover a broader audience of developers. Also, an author might decide to keep a model private while developing it, and make it public once finished. Thus retaining control over the development process.

3.5. Version Control

It is easy to think in version control functionalities (e.g. for models) as adequate for the platform. However, at this point, such functionality will not be considered. Firstly, implementation of version control functionalities is known as a hard task (France et al., 2006a). Then, models can be described in many languages (some of

them domain, community or research group specific), which results in known versioning problems (France and Rumpe, 2007). By merging these two factors we face a complex problem that we decided not address at the moment. Furthermore we are more interested in cataloging models (where the models should be more stable and ready to be used by other users), than in a centralized development tool as is the case of control version systems. Instead of managing version control, the platform should support users in performing version control themselves, allowing them to manually register new versions of the models. These versions are to be sequentially numbered.

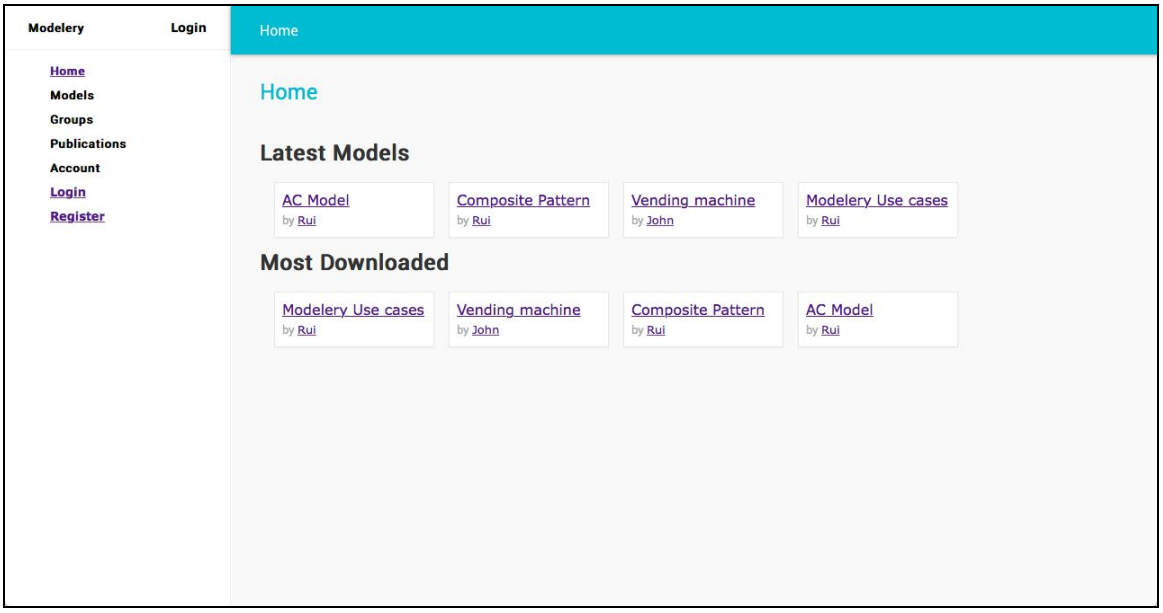

Figure 1 The Modelery's main page.

4. The Modelery

In order to answer the above requirements we have developed the Models Refinery (Modelery)[1] platform. Our platform combines the proposed functionalities in a Web environment accessible through the browser as depicted in Figure 1. Additionally, it offers a set of web services for supporting interoperability and integration with external modelling environments. The platform was developed according to a model driven methodology, and used the Modelery itself to keep track of the source models.

[1] http://modelery.di.uminho.pt

4.1. Artifacts repository

The models (artifacts) repository functionality was the major concern in designing and developing the platform. The Modelery archives and makes available, not only the models, but also their meta-data. This meta-data (see Table 1) constitutes the model's entry, provided by the user when submitting to the repository.

Item	**Description**
Name	The name of the artifact
Author	The author of the artifact, automatically associated
Date	Date of submission
Description	A description of the artifact
Institution	Institution where the artifact was produced
Tool	Tool which originated this artifact
Tags	A set of tags, associated with the artifact
Language	The language in which the artifact was created (for instance, programming language)
Publications	List of publications associated with the artifact
Visibility	Visibility of the artifact: Only to author, to group, or public
Updateable	Whom may update the artifact: only the author, or the group
Group	The group which the artifact may belong
Image	An image representing the artifact
File	The artifact file itself

Table 1 Artifact Meta-data.

While any user might search and view (public) models, registration is required in order to create a new one. Figure 2 presents the user interface for adding a model. Mandatory fields are signaled with an asterix ("*"). Hence, for example, a model must always have a name and an author. The model file must be also specified, and it is then uploaded and stored online in the platform. The model's author is able to both update the model (by submitting a new version - with the previous version being kept on record), and to edit the models' meta-data.

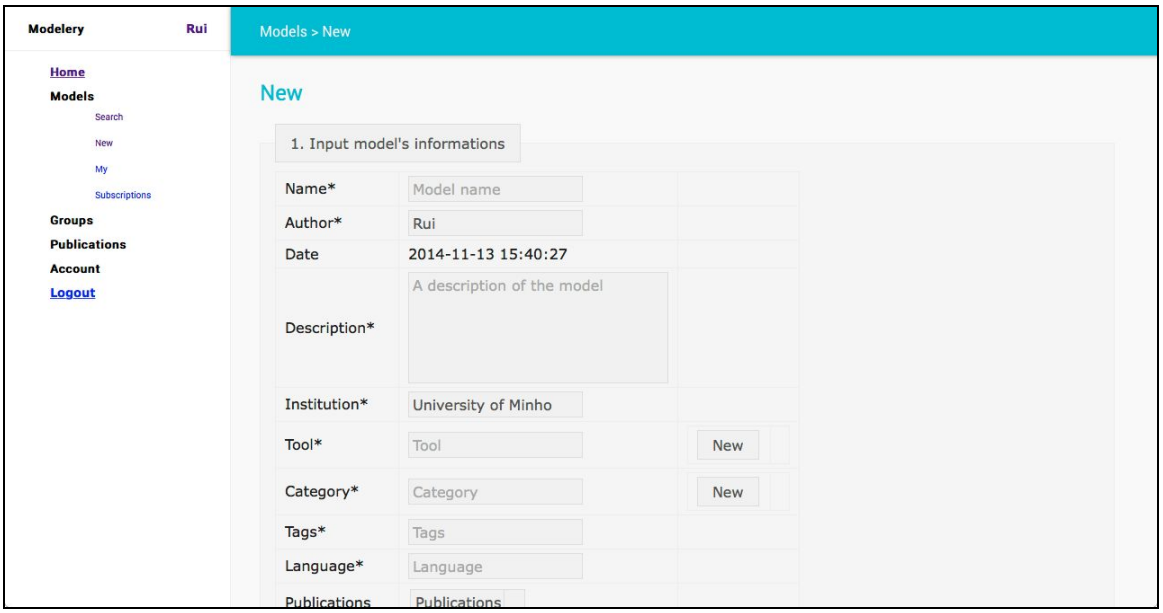

Figure 2 Adding a model.

In accordance with the identified requirements, the platform supports a number of features that help manage and share models: groups, publications and visibility options.

Two complementary ways to specify the context of a model are provided. Firstly, a model can be part of a group. This possibility enables us, not only to aggregate a set of models in a specific group, allowing for their categorization, but also to restrict access to a set of persons which may view or update them, the members of the group. Secondly, the platform provides also a means to identify the publications in which a model is involved. This constitutes a further dimension through which to classify and access models.

A visibility level can be defined for each model. The visibility level defines if the artifact is visible to everyone, visible to the group members, or visible only to the author. Besides visibility levels, the platform supports also the definition of which users might update the model. Here, the owner of a model may let a group update it,

or restrict updates to himself/herself. The visibility level and who may update a model are independent properties, since the model may be visible to the group, but only the author might have permission to update it.

Once the models have been added, they can be searched for. By selecting the search option, a listing of the existent models is presented, as depicted in Figure 3. The user may then input some text, and the listing will be filtered according to the search criteria, presenting only those models whose name or description match the text being input.

Because models can evolve over time, a user might wish to follow the progress of a specific model he/she has found or added to the platform. In order to ease the user's access to those relevant models, the platform implements a model "tracking" functionality. Users they provided with a list of references to the models they have chosen to follow. Other functionalities aimed at providing an overview of the state of the repository include a dynamic main page, which presents information such as the last submitted models and most downloaded models, and a tag cloud. This provides an overview of the contents of the repository, emphasizing most relevant models.

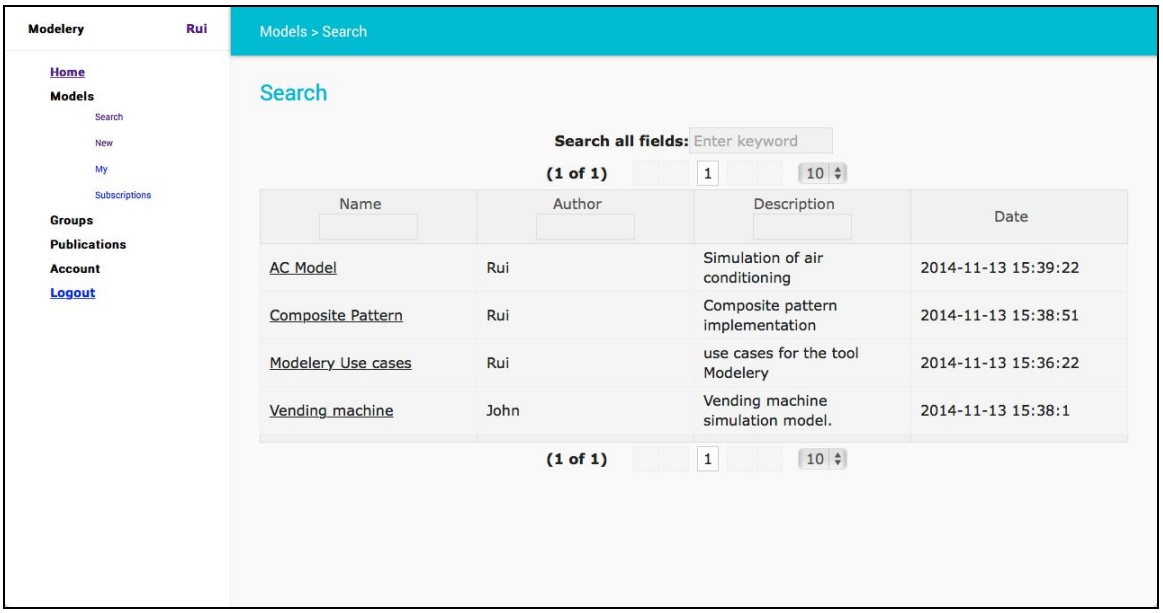

Figure 3 Searching for a model.

4.2. Publications management

As mentioned above, the Modelery supports the management of publication entries. The publications are registered with their name, abstract and URL for the article location, as shown in Figure 4. Contrary to what is provided for models, publications

management does not support uploading the publication itself into the platform. We consider this to be a more efficient approach, as the platform's focus is not the publications themselves. Since publications may have more than one author, they are not automatically associated with the user which created them. Information of the authors is in the publication document itself. Publications' data can be input manually or obtained from a DOI. The information can be exported to LaTeX.

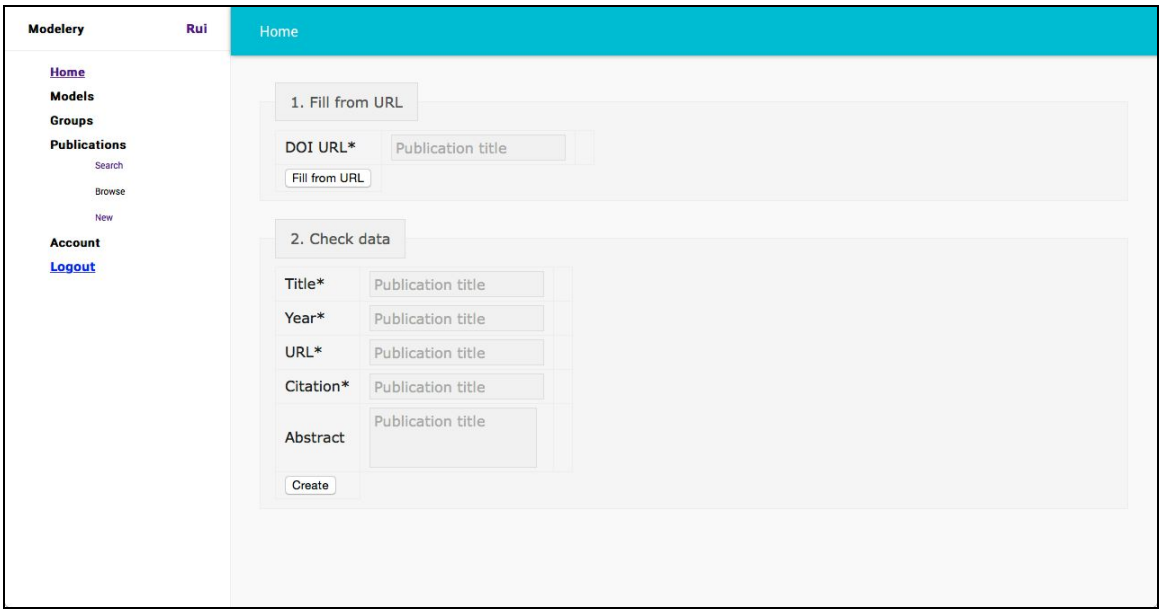

Figure 4 Adding a publication.

The relation between the models and the publications can be explored starting from different dimensions in the repository. On the one hand, publications may refer a specific model or list of models, and it is possible to list the models associated with a publication. On the other hand, a model may be referred to in multiple publications, and it is possible to view all its associated publications. This functionality provides a convenient way to explore publications along with models, and at the same time provides more information for a given model. The same is also true for the tools, i.e., view the tool associated with a model, or the models associated with a tool. It allows also exploration of the support material (i.e. the models) of the publications. Besides this browsing facilities the textual search functionality is also provided for publications.

4.3. Collaborative functionalities

Collaborative functionalities are achieved by using Web 2.0 functionalities to promote interaction among users (Pérez et al., 2012). This is achieved through a number of means. Users are automatically associated with any group, model,

comment or update that they create. This allows other users to know who is the author of a given model, or the owner of a specific group.

A functionality that is essential for promote collaborative behaviors is the possibility of users to exchange messages inside the platform. The Modelery supports both personal one-to-one messages, and more public messages in the groups. The groups have a forum like message system which can be either public or private. Finally, it is also possible to comment the models.

Interaction between the users is also supported through the models in the platform. Registered users may interact with a model by adding comments (for example, suggesting improvements, which will fosters the evolution of the models). Additionally, users might rate models on a 1-5 scale, thues expressing their assessment of the models.

4.4. Implementation

The Modelery was developed according to a multi-layer architecture, using a model driven approach. The business layer is composed by three main parts: the model (repository), which includes the models and all the related information; the user, which handles user related data, such as accounts; and the groups, which supports the groups (forum) functionalities. The Modelery class diagram is shown in Figure 5. The persistence is achieved through the Hibernate framework, plus MySQL database.

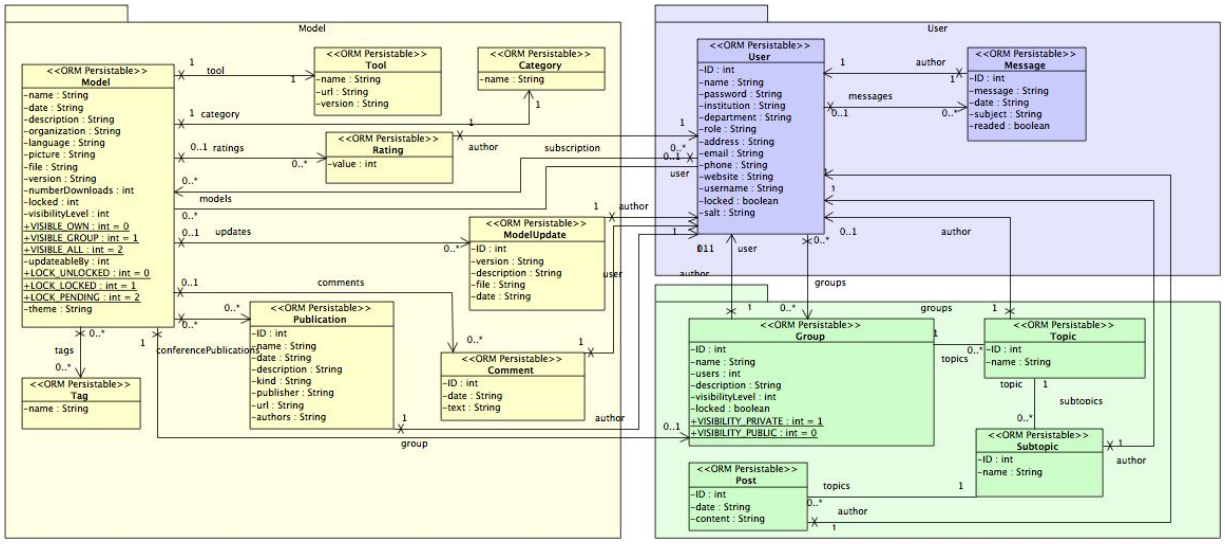

Figure 5 Modelery business layer class diagram.

The presentation layer was initially implemented using Java Server Pages (JSP) and servlets over the business layer. Due to the relevance of usability considerations for the platform's success, an effort was made to create a responsive user interface (for instance, avoiding full page reloads for small requests) in order to improve the experience of the users. In a first iteration of the platform this was mainly achieved resorting to Ajax (Zakas et al., 2006), by performing modular page loadings. This also enabled us to provide more lightweight Web pages and reduced bandwidth usage. Resorting to a combination of HTML5 (Crowther et al., 2014), Cascading Style Sheets version 3 (CSS3) and jQuery, we are able to improve the user interface by, for instance, providing early error detection when filling fields in the Web page, and better feedback (including animations when performing changes to the page contents). In the second iteration the usage of Primefaces[2] components with Java Server Faces has contributed to a more responsive and efficient interface.

Additionally, the Web interface was developed according to Responsive Web Design (Ethan, 2011) concerns, thus taking into consideration compatibility with old browsers. Even if the visual aspect is not kept (mainly due to lack of CSS3 compatibility) all the functionalities remain usable.

Following a multi-layer approach enables improvements or changes to specific platform components with minimal or no impact on the others. Such was the case in the second version of the platform, were the Java Server Faces (JSF) framework

[2] http://primefaces.org/ (visited November 7, 2014).

replaced JSP (at the user interface level), and a set of web services were added (see the next section).

4.5. Interoperability

For all its benefits, using a Web-based repository means using a an additional system. Storing, loading and updating models might be easier to do inside the applications used for developing the models themselves. With that in mind, in the second iteration of the platform set of REST web services were developed, based on JSON (Javascript Object Notation), to allow other applications to interact with the Modelery. The Modelery's multi-layer architecture eased the integration of the web services component. A servlet was developed which handles the HTTP POST requests.

Web services are grouped according to the business entities: groups, models and publications. For each, there is a class which handle the specific requests, with each method in the class corresponding to a specific web service. The servlet then forwards the requests to the corresponding classes and methods. For this to work, which class and method is requested must be specified in the POST message. Alternatively, we could have created a servlet for each request type, but such would have increased the complexity of the solution.

Figure 6 shows an example of an HTTP POST request and corresponding JSON response. At the top, the architectures of the Modelery and uCat (an external application, see Section 5.2) are shown. The Modelery web services allow communication with uCat, via the Modelery Connect and HTTP. The HTTP request is shown below, where it is possible to see the several web service parameters such as the method and class. Also, at the bottom the corresponding response for the given request (e.g. a model entry) is shown.

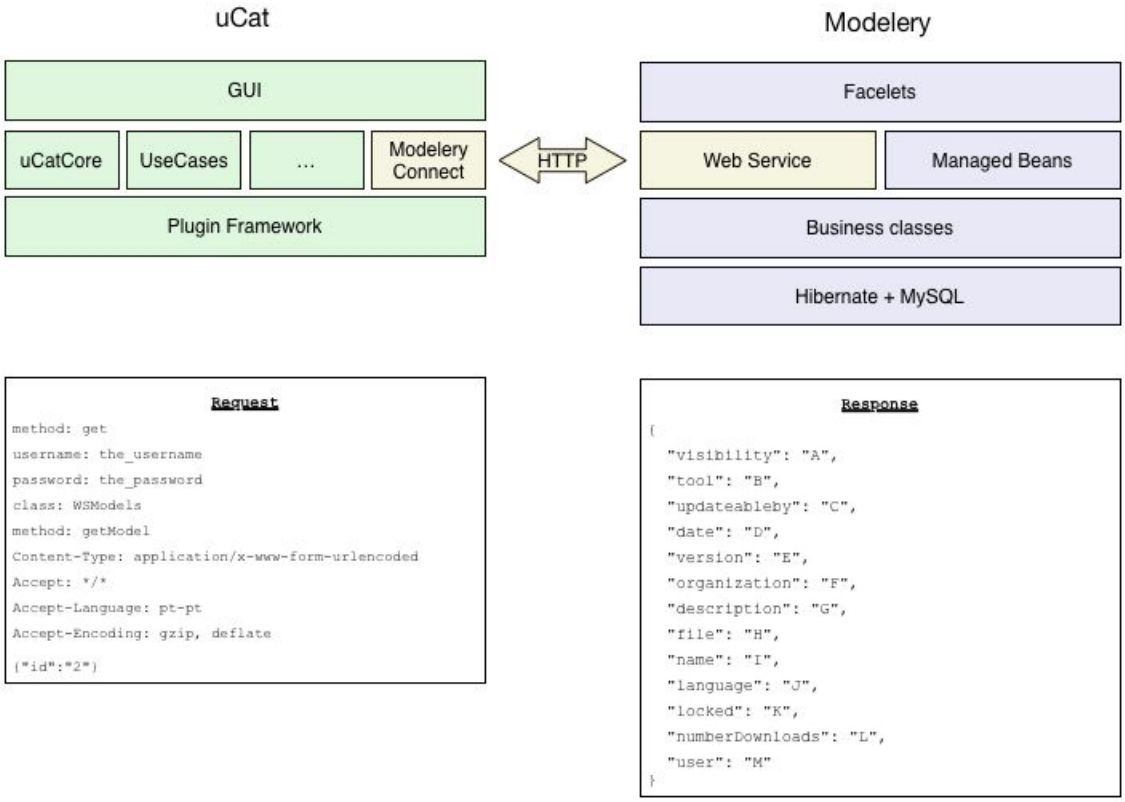

Figure 6 Architecture of Modelery and uCat, and resective json Response and HTTP POST request.

The web services are meant to be used as an integration of the Modelery core functionalities in third party applications. Hence, we consider that some functionalities, such as creating user accounts should be left in the web page itself.

The list of available web services is:

- List artifacts: Allows to list artifacts, filtered by the tool which originated them or by name;
- Get an artifact: Allows to retrieve all the information related with an artifact;
- Create an artifact: Allows to create a new artifact;
- List tools: List the tools existing in the Modelery;
- Create a tool: Add a new tool;
- List the categories: List existing categories;
- List the group: List existing groups;

- List the languages: List existing languages;
- List the publications: List submitted publications;
- Create an update: Add an update to a model;
- Get a model update: Get a given model update.

This list of web services is enough to support interaction with other applications, as we show in the next section.

Alongside the web services a Java library, the ModeleryConnect, was developed which creates an abstraction layer over the usage of the web services by providing methods that corresponding to the above described functionalities.

5. Applications

This section describes two examples of use of the Modelery platform. In one case, the platform was used to replace an existing repository, the main interest being to provide access to models developed by the team and external collaborators in the specific topic of Human Computer Interaction (HCI). The other case, illustrates a concrete example of the integration of repository functionalities, via the ModeleryConnect library, into our own tools. With this it was possible to further analyze how well the web services allow an integration of the tool with the Modelery.

5.1. HCIspecs repository

The use of models to reason about interactive computing systems or Human Computer Interaction is an active field of research with different modelling languages and tools being used (see Bolton et al., 2013 for a review of the area). HCIspecs is a repository focussed specifically on this type of models. It grew out of a need to make available models in such a way that they could be easily shared and referenced to (for example in publications). The goal was also to make it available to the community at large.

The first version of the platform presented models organized by tool and by paper. However, that fact that it was implemented on top a general purpose Web content management system (phpwcms[3]) meant that a very specific approach to adding content had to be devised so that the end result was the one intended. Despite the

[3] http://www.phpwcms.de (last visited 5/12/2014)

platform's qualities, achieving the intended result meant using it in ways it had not been designed to. The end result was that adding models and papers to the platform was a non trivial process making it hard to maintain the platform and unrealistic to provide writing access to other users.

Adopting the Modelery as the new platform for HCIspecs is simply a matter of installing the platform and migrating the models. By adopting the modelery we immediately gained the possibility of enabling others to add models to the platform Additionally we gained the possibility of supporting discussions on the models, fostering interaction between the community. We are currently in the process of migrating the models from the previous platform to the new one.

Additionally, we have added the capability of directly adding models to the repository from our modelling tools. The next section discusses one such case.

5.2. Use Cases Analysis Tool

The Use Cases Analysis Tool (uCat) is a tool to support automatic data extraction from use case specifications (Couto et al., 2014b). Usage of the tool starts with the input of use case specifications. Such specifications are then translated into OWL, making it possible to perform data inference on the Use Case, namely requirements pattern inference. Such patterns enable the automatic generation of the architecture of software prototypes for the described system. In uCat use case are input as descriptions (persisted as XML files). Such files are the models, which we wish to store in the Modelery. We have integrated the uCat tool with the Modelery by integrating the developed Java library in the tool, in order to provide model registration and search functionalities. Figure 7 illustrates adding a model to the repository. In the figure it is possible to see the several required field. At the top, the user should specify the Modelery username and password. Next, the user should provide the Modelery web service URL and the model metadata details.

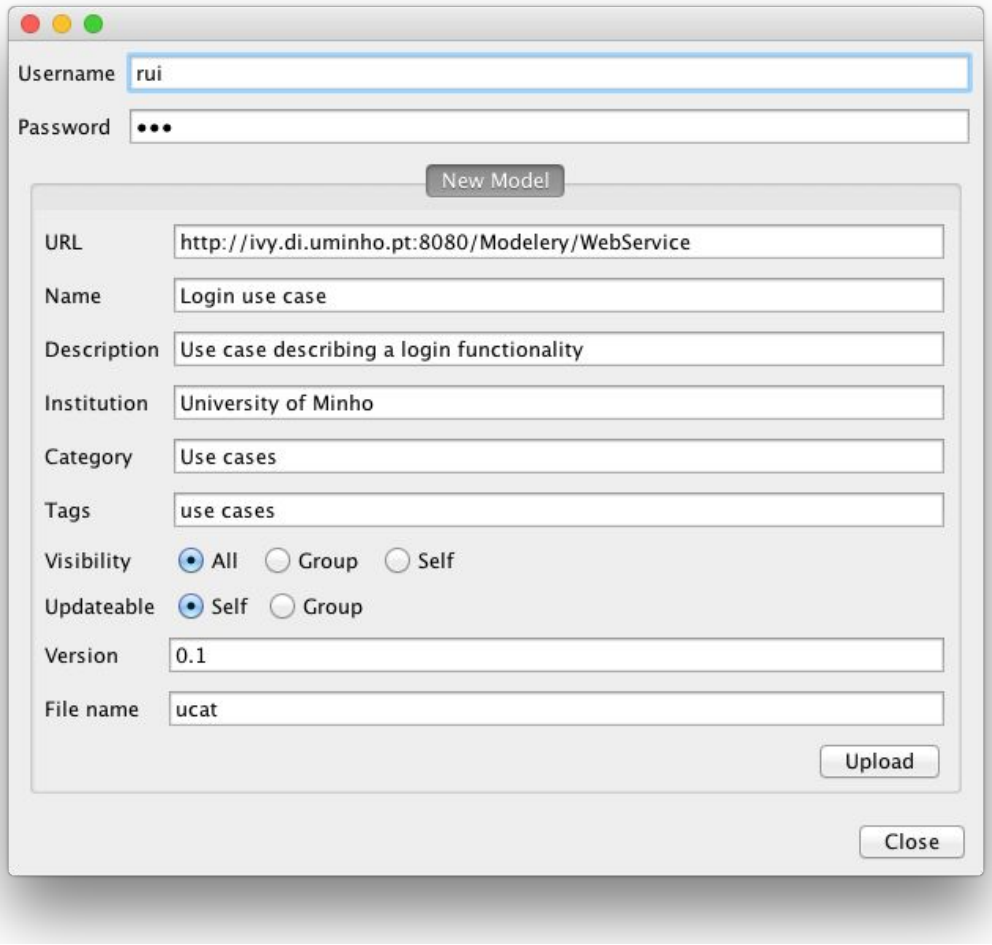

Figure 7 Uploading the login use case in uCat.

Once the models are uploaded, they can be seen and interacted through the Web interface as any other model. It is also then possible to list and download the models in the Modelery from inside uCat, as presented in Figure 8. This functionality takes advantage of the web service's support to listing the models which match a given tool only (uCat, in this case). In the left hand side of the figure, it is possible to see the previously uploaded Login use case. When a model is selected, its details are presented (see right hand side window) and it is possible to select a specific version to download. In this case it is possible to see that we have only the base version of the model.

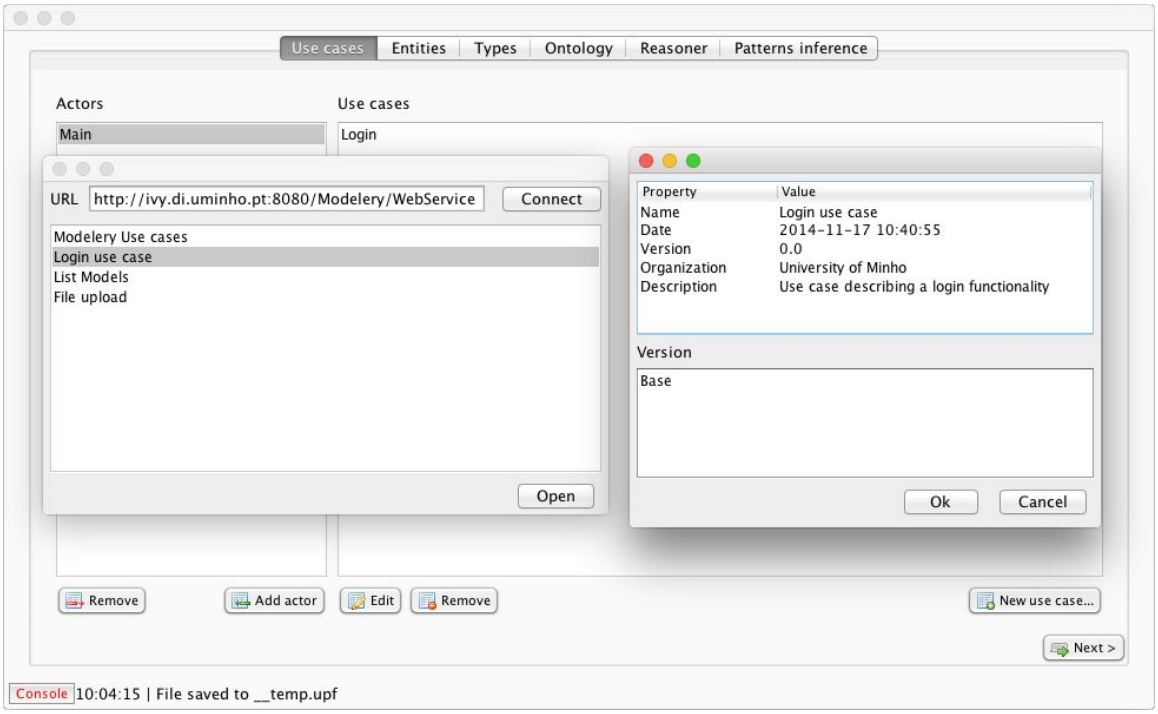

Figure 8 Listing and downloading a model from the Modelery.

Next we create another use case scenario, (for instance, a logout functionality). As the model was downloaded from the Modelery, further uploads must be done as updates. Again this can be done from inside uCat. Figure 9 shows the interface to upload a new version of the model. We introduce the new version code and a short description, and upload it.

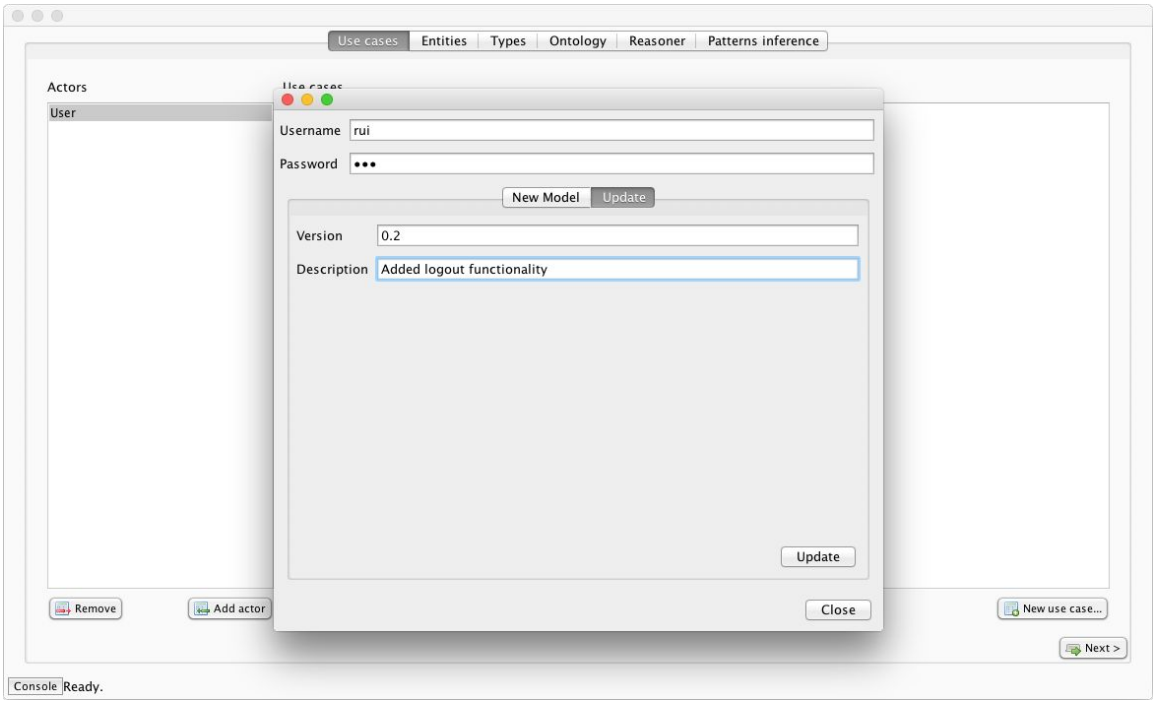

Figure 9 Adding a new version to the Login use case.

Our model has now two versions (the base, and the version with the logout functionality). If a model has several versions, it is possible to list them and download a specific one. In Figure 10 it is possible to see that now we have both the 0.2 and the base versions.

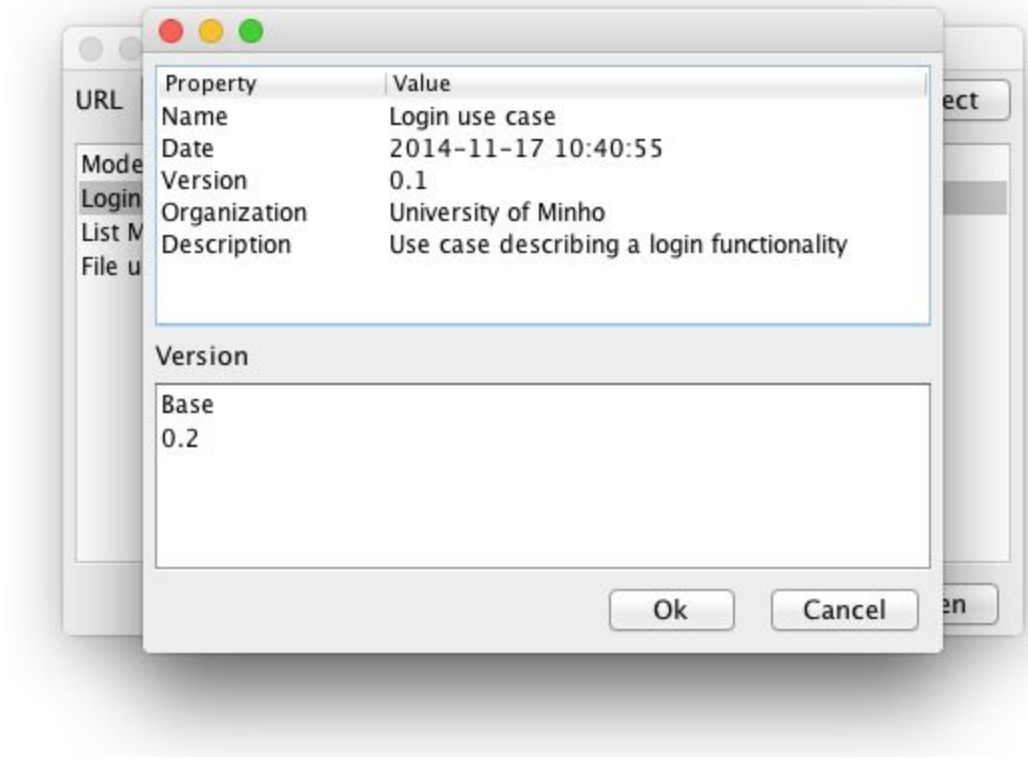

Figure 10 Downloading the version 0.2 of the use case from the Modelery.

6. Discussion

The Modelery is now a fully functional platform, which we consider implements the more relevant functionalities identified in Section 3.

An alternative approach to achieve a similar platform would have been to conjugate several other platforms into a single environment. For instance, a Concurrent Version System (CVS) (such as SVN or GIT) for models' management, along with an online forum (such as phpbb) for discussion issues. However, the approach taken presents advantages over the integration of multiple platforms. First, CVS system are mainly used and optimized for textual documents (such as source code). They lack model targeted functionalities, and it is harder to add functionalities (such as an online model editor) later on. Furthermore, CVS systems are not targeted for sharing and cataloging. Using an online forum for our objectives suffers from similar issues as the usage of a CVS for the models, with the inability to provide specific functionalities. Integrating visibility levels in a CVS, or groups, managed by the users, in the forum, would have been a very hard and time consuming tasks. Combining these functionalities to collaborate together, by providing a platform as

coherent and as practical as ours would have been more costly than developing this one. Finally, a poor integration of these technologies might easily lead to an unpractical platform, and result in a project failure.

Some of the repositories discussed in Section 2 offer online models' editing. That is an interesting functionality. However, not suitable for our repository at the moment. Since we allow any kind of model in our repository, supporting editing functionalities would require either a restriction on the type of supported models (by imposing a metamodel, for instance), or selecting a subset of models for online editing support. We have chosen to ignore this functionality for now, since it would not lead to a solid and robust editor.

Comparing our platform against other repositories, it is possible to draw some conclusions. There are some similarities between our tool and ReMoDD, since our objectives are somehow similar. However, we provide some improvements with The Modelery. First, our platform provides a larger group of functionalities without requiring registration. An unregistered user is free to explore all the public information, from groups to models and publications. ReMoDD is considerably more restricted in model browsing. The only way to search content in the site (any kind of content) is by textual search. Another possibility is to list all of the models. The platform provides also a forum, however completely disconnected from the models. Finally, it provides a workshop catalog system, once again, disconnected from the models. Viewing a model's information is very limited, since only few informations are displayed. ReMoDD claims to be a repository for model driven development, however our platform might provide a better support for model driven methodologies by overcoming some of ReMoDD shortcomings.

ECOBAS has different purposes, being aimed at a specific area and focusing on modelling and simulation. In what concerns management of models, ECOBAS is somewhat limited in terms of the search functionality, since it only supports the listing of models by name, or performing a textual search. Opening a model's entry provides a large amount of information, but lacks some of the details we consider relevant, such as a visual representation of the model or the author. ECOBAS lacks also other functionalities such as publications management and discussion groups. From this point of view, the Modelery provides a more complete environment as a model repository.

The Apromore platform shares some of our objectives, but it is currently in a preliminary phase of development. The platform allows public models' submission only, limiting the models' scope. The model entries do not provide very complete

information, since apart from its name, it is only possible to view their language, domain, ranking, version and author. The platform offers an interesting online model editor. However that editor is language specific, allowing only to edit one kind of model. Also, Apromore provides no other functionalities than a model repository. At the moment, this platform has limited browser support. Modelery provides a more usable option, since it is ready for use. Users are free to register (contrary to Apromore), and submit any model, as well as their relevant information.

Table 2 summarizes the comparison of the platforms.

Tool	Fully Web	List	View	Comments	Download	Public access	Groups	Advanced Search	Open platform	Software oriented
ReMoDD	✔	✔	☐	✔	✔	✖	☐	☐	✖	✔
ECOBAS	✖	✔	☐	✖	☐	✔	✖	✔	✔	✖
Apromore (prev.)	✔	✔	✔	✖	✖	✔	✖	✔	✔	✖
Shelfari	✔	✖	✔	✔	✖	✔	✔	✔	✔	✖
SRI	✔	✔	✔	✖	✔	✖	✖	✖	☐	✔
GenMyModel	✔	✖	✔	☐	✖	☐	✖	✖	☐	☐
Modelery	✔	✔	✔	✔	✔	✔	✔	✔	☐	✔

Table 2 Comparison of the analyzed repositories.

7. Conclusions

In this paper we have described a collaborative repository for software artifacts, with a special focus on models, patterns and catalogs. We presented the Modelery, a platform which combines an online artifact repository, publication management and collaboration functionalities. The presented functionalities came mainly from our needs to store, manage, catalog and make the artifacts we produce during our

research projects, available online. Also, with this platform we have created a new means to discuss the artifacts within discussion groups. After experimenting with a first version of the platform (Couto et al., 2014a), we have introduced to major improvements. Firstly, we have used Java Server Faces (JSF) to improve the interaction with the user. Secondly, we have provided a set of web services to support connectivity of the platform with other tools.

We are now using the repository for our own needs. In the longer run we consider the possibility to include other functionalities in the platform. Namely, the possibility of integrating editors or the generation of graphical representations for particular modelling languages, and also integration with verification and validation tools (e.g. for certification purposes). The addition of web services to the platform allows to open new horizons. We are considering the possibility to develop standalone applications for certain functionalities, such as a desktop application for keeping some models locally. In the same line, we are also considering further improving the web services with more functionalities.

Acknowledgments

This work was carried out in the context of project Languages And Tools for Critical rEal-time Systems (Ref. NORTE-07-0124-FEDER-000062), financed by the North Portugal Regional Operational Programme (ON.2 - O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF), and by national funds, through the Portuguese funding agency, Fundação para a Ciência e a Tecnologia (FCT).

References

Bernstein, P. A. and Dayal, U. (1994) An Overview of Repository Technology. In Proceedings of the 20th International Conference on Very Large Data Bases, San Francisco, CA, USA, pp. 705–713.

Bolton, M. L., Bass, E. and Siminiceanu, R. (2013) Using formal verification to valuate human-automation interaction, a review. In IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, no. 99, pp. 1–16.

Brosch, P., Langer, P., Seidl, M., Wieland, K. and Wimmer, M. (2010) Colex: a web-based collaborative conflict lexicon. In Proceedings of the 1st International Workshop on Model Comparison in Practice, New York, NY, USA, pp. 42–49.

Campos, J. C. and Harrison M. D. (2009) Interaction engineering using the IVY tool, in ACM Symposium on Engineering Interactive Computing Systems (EICS 2009), New York, NY, USA, pp. 35–44.

Campos, J. C., Saraiva, J., Silva, C., and Silva, J. C. (2012) GUIsurfer: A Reverse Engineering Framework for User Interface Software. In Reverse Engineering -Recent Advances and Applications, A. C. Telea, Ed. InTech, pp. 31–54.

Cavalcanti, M. C., Mattoso, M., Campos, M. L., Llirbat, F. and Simon, E. (2002) Sharing scientific models in environmental applications. In Proceedings of the 2002 ACM symposium on Applied computing, New York, NY, USA, pp. 453–457.

Couto, R., Ribeiro, A. N., and Campos, J. C. (2012) A Patterns Based Reverse Engineering Approach for Java Source Code. In Software Engineering Workshop (SEW), 2012 35th Annual IEEE, pp. 140–147.

Couto, R., Ribeiro, A., Campos, J. (2014a) The Modelery: A Collaborative Web Based Repository. In Computational Science and Its Applications – ICCSA 2014, vol. 8584, B. Murgante, S. Misra, A. C. Rocha, C. Torre, J. Rocha, M. Falcão, D. Taniar, B. Apduhan, and O. Gervasi, Eds. Springer International Publishing, pp. 1–16.

Couto, R., Ribeiro, A. N. and Campos, J. C. (2014b) Application of Ontologies in Identifying Requirements Patterns in Use Cases. In Proceedings 11th International Workshop on Formal Engineering approaches to Software Components and Architectures, FESCA 2014, Grenoble, France, pp. 62–76.

Crowther, R., Lennon, J., Blue, A. and Wanish, G. (2014) HTML5 in Action. Manning.

Dirix, M., Muller, A. and Aranega, V. (2013) GenMyModel: An Online UML Case Tool. In Joint Proceedings of Tools, Demos & Posters: 14.

Do, H., Elbaum, S. and Rothermel, G. (2005) Supporting Controlled Experimentation with Testing Techniques: An Infrastructure and its Potential Impact. In Empirical Softw. Engg. 10, 4, pp. 405-435.

Dolk, D. R. and Konsynski, B. R. (1984) Knowledge Representation for Model Management Systems. In Softw. Eng. IEEE Trans. On, vol. SE-10, no. 6, pp. 619 –628.

Ethan, M. (2011) Responsive Web Design. A Book Apart.

France, R., Bieman, J. and Cheng, B. (2006a) CRI: Collaborative Project: Repository for Model Driven Development (ReMoDD), Colorado. State University.

France, R., Bieman, J. and Cheng, B. (2006b) Repository for model driven development (ReMoDD). In Proceedings of the 2006 international conference on Models in software engineering, Berlin, Heidelberg, pp. 311–317.

France, R. and Rumpe, B. (2007) Model-driven Development of Complex Software: A Research Roadmap. In 2007 Future of Software Engineering, Washington, DC, USA, pp. 37–54.

Jouaulta, F., Allilairea, F., Bézivina, J. and Kurtevb, I. (2006) ATL: A model transformation tool. In Science of Computer Programming, Volume 72, Issues 1–2, 1, pp. 31-39.

Karagiannis, D. and Kühn, H. (2002) Metamodelling Platforms. In Proceedings of the Third International Conference on E-Commerce and Web Technologies, London, UK, UK, p. 182–.

OMG (2014) XML Metadata Interchange (XMI) Specification.

Pérez, R., Benito, B. M. and Bonet, F. J. (2012) ModeleR: An environmental model repository as knowledge base for experts. In Expert Syst Appl, vol. 39, no. 9, pp. 8396–8411.

Rosa, M., Reijers, H. A., van der Aalst, W. M. P., Dijkman, R. M., Mendling, J., Dumas, M. and García-Bañuelos, L. (2011) APROMORE: An advanced process model repository. In Expert Syst Appl, vol. 38, no. 6, pp. 7029–7040.

Wang, H., Johnson, A., Zhang, H. and Liang, S. (2010) Towards a collaborative modeling and simulation platform on the Internet. In Adv Eng Inf., vol. 24, no. 2, pp. 208–218.

Zakas, N., McPeak, J. and Fawcett, J. (2006) Professional Ajax. Wrox.