Introducing Useme: the Open Requirements Management Tool

Poster presented at EclipseCon 2008

Barbara Rosi-Schwartz & Joel Rosi-Schwartz
Etish Limited

Further information about Useme and about us may be found on the Useme website and on the Etish website
The main features

- Useme is an open source collection of tools that together form an automated requirements management system.

- Useme is use cases driven and it merges the two concepts of UML model and document specification by enforcing a well organised, structured definition of the documents in the first instance and by automatically updating all relationships between the documents when requirements evolution or change dictates it.

- Useme utilises specialised, custom built form based editors with one type of editor for each supported type of requirements document.

- Useme supports rich text, including text that may contain smart links.

- Useme supports the sharing of production and consumption of requirements documents among all the different stakeholders in a software development team.

- Web based publishing tool which ensures that all artifacts are published for consumption among all the stakeholders of the development project. The publisher ensures that all documents exhibit the same look and feel and renders all smart links and special elements in the documents in a fashion appropriate for the output type.
Because of issues like these

- The affordable requirements related tools ultimately rely on the definition of a requirements document via a free form text editor.

- Even utilising document templates to aid document standardisation, the lack of a well defined and enforceable structure in the textual documents makes the task of standardisation quite difficult.

- Most organisations that are committed to UML as a common language for team communication often experience a gap between the UML requirements model and the textual specification documents.

- Even in cases where the most important of the relationships are captured in the textual documentation, the task of maintaining them by hand through requirements evolution and change is daunting.

- Except for high end tools, most other automated systems are single user, making it difficult for team members to work on requirements documents as a community and for all the project stakeholders in general to share the information expressed in these documents.
The Future

Useme growing taller

- Full support for a Glossary document.
- Full support for a Software Requirements Specification document.
- Enablement and management at both the document and at the project level of:
  - notes
  - issues
  - functional & non functional requirements
- addition of reporting and validating capabilities.
- and much, much more....

With ORMF providing the foundation

- A structured model for textual requirements specification documents.
- The enhancement of smart links.
- General availability of common requirements related elements.
- The availability of complex views and reports.
- Provide a rich infrastructure of common patterns, services and mechanisms.
- Support for versioning and change control.
- Data import and export facilities.

More than a tool

Our vision goes well beyond version 1.0. The next steps include not only an improved tool but a framework, the ORMF.

The goal of the Open Requirements Management Framework (ORMF) is to extend the Eclipse platform to create an open source framework for building Eclipse based tools that manage requirements documents.

ORMF will be a solid multi-tier framework that will enable the Eclipse platform to support creation, development and management of disparate types of requirements documents by utilising a common set of services, patterns and mechanisms.

If you don’t know where you’re going, any road will take you there. (Anonymous)
Would you like to contribute?

That’s fabulous, there are many ways to do it

- Give the team feedback on your experience of using Useme; we need your reactions, both positive and negative to understand the needs of the community.
- Found a bug, then please file a bug report (or, even better, a patch;-)
- If you have ideas for improvements or new features, please file a feature request.
- Fancy yourself an expert in Requirements Management, please contribute your expertise and guidance.
- Want to get your hands dirty, come build with us; whether you are an architect, analyst, designer, developer, documenter or tester, your skills can help.
- We would love to partner with others.
- Corporate sponsors are very much welcome.
- Simply want to drop a few coins in the tin cup, we won’t blush :-)
- Last, but certainly not least, hire Etish Limited. It is this income that allows us to continue developing Useme and the ORMF.
The Useme Servers

- A server is a repository of Useme projects. As such, it centrally stores all requirements documentation, which it serves to both Useme plug-in based editor clients for document production and to Useme Publishers for document consumption.
- It is installed, configured and managed by the Useme Administrator.
- It is completely defined, at the client level, by its connection parameters and it is reached via the user’s authentication parameters. These are provided in the Useme client once and only once.

The Useme Projects

- A project is a container for all requirements documentation for a given software project.
- Projects are stored on Useme servers but manipulated by the Useme client plug-ins.
- An Analyst may only work on a project if she has been granted authorisation by the Useme Administrator.
- A project may contain folders, UML packages and various types of requirements documents. Folders are containers of documents and, in specific cases, of UML packages. UML packages contain Use Case and Actor specification documents. They may also contain other UML
Useme documents are organised as XML document and underlined by a specific XML schema (one per type of document). The schema represents the significant elements that compose a particular requirements document.

**Why structured?**

- The structured nature of the documents facilitates automatic control and validation of the content as performed by Useme.
- Structured documents, coupled with the guidance and support provided by Useme, enable the definition of requirements in a more exact fashion, but without loss of communicability to the end users of the project.
- Being structured, the documents are consistent across an entire project and facilitate sharing.
- The availability of structured documents enables tasks other than content creation, such as sophisticated searches and reporting across all the documentation set or test analysis. All of this can be automated.
Relationships

- With Useme it is easy to define UML relationship in a document. The tool guides the Analyst through the correct definition of relationships. Behind the scenes, relationships are defined in all of their UML features, namely stereotype, end points, multiplicity and navigability.

- Each document specific plug-in contributes those stereotypes that are unique for the type of requirements document in question (ex. “include” or “use subflow” are contributed by the use case editor).

Smart links

- When expressed within a textual context, relationships are referred to as smart links. This is perhaps the most innovative feature of Useme.

- Smart links (as well as relationships) are automatically maintained and updated by Useme during the project life cycle through changes in either one of the two relationship’s participants. Elements that participate in a link cannot be deleted.

- Smart links are navigable in the published document, so that it is straightforward to follow them through the documentation.
The collaborative nature of Useme requires mechanisms for all users to be kept up to date with the latest changes to a project.

**Possible strategies**

- Rely on the user to explicitly refresh when desired. This is inappropriate because information cannot be kept up to date by the user alone.
- Force a refresh every time a given project is updated on the server. This would undesirable because, in a truly collaborative environment, it would cripple the system.

**Our solution**

- The project is automatically and silently refreshed every time the user request an operation that requires server communication. If the refresh invalidates the operation requested, the user is informed and the operation is aborted.
- A periodic refresh mechanism refreshes all currently open projects one the user’s workbench. The frequency of the refresh depends on the context: a periodic refresh happens more frequently when a wizard is in focus than when the main workbench is displayed.
Adding a server location

The first thing you do is provide Useme with the parameters that define the server on which the Useme project you want to modify is located. You invoke the New Server Location wizard, which is shown in the screenshot below.

Having completed the operation, the Useme perspective will be instantiated.
Viewing the Useme perspective

The perspective contains, on the left, the Servers view, for interacting with servers, and the Projects view (shown), for manipulating projects.

The editors on the right are structured multipage form editors. The one is the use case editor, with the Alternate Flows page currently in focus.
Creating a document

You create documents by invoking the right wizard for the desired type. Below is a New Use Case wizard. The wizard is initialised with the selection in the Projects view and it performs all needed validation. After creation, the document is automatically opened in its appropriate editor.
Adding a relationship

You add a relationship between two documents via a wizard invoked from the editor’s Relationships page. After you have defined it, the relationship as in the list below. Note that the icon warns that the relationship has been added, but it has not yet been utilised in any textual context in the
Adding a smart link

You do this by positioning your cursor in a smart text field and invoking Smart Link wizard (see below). You define the smart link’s stereotype and wizard shows you the possible choices. Below we have a smart link of type Include and therefore the previously defined relationship is shown as a
Viewing smart links

Having added the smart link, it will appear in line at the location you chose shown below. The smart link is a read only block of text shown in red font. If the referenced document changes name or location, the smart link below will be automatically updated to reflect the change.
Keeping everything synchronised

Useme ensures that your view of a project is kept up to date with the content on the server. In this example Useme warns you that a use case with the same name as the one you have typed into the New Use Case wizard has been added to the same parent while your wizard is up. The wizard is in
Publishing!

Nothing much for you to do here! As soon as your document is saved, it is available for publication. UseMe produces the document upon request, all of its changeable components guaranteed to be up to date. Below is an example of a use case document published in HTML format.

UseMe

Use Case Specification: Rename document
Version 1.1

1 Description
1.2 Purpose
To change the name of a document.
1.3 Type
System
1.4 Hierarchy
Parent
1.5 Status
Draft
1.6 Priority
Medium
1.7 Brief Description
Once a document has been created with a given name in a Captive project, it can be renamed via this use case. Only one document may be renamed at a time. At the moment we do not allow in-place renaming in the Projects view and therefore the rename operation is performed via a wizard. However in future releases we may consider allowing in-place operations as well. This use case describes how the renaming is achieved.

It is to be noted that both moving and renaming are considered refactoring operations.

2 Relationships
2.1 Primary Actors
2.2 Secondary Actors
2.3 Generalizers

2.5 Includes
- `rename reflesct meta data`
- `rename reflesct meta data`
- `rename reflesct meta data`

2.6 Is Included By
2.7 Extends
2.8 Is Extended By
2.9 Realizes
2.10 Is Realized By

3 Pre-Conditions
3.1 Document selected
One and only one document must be selected in the Projects view.
3.2 Document not locked by another user
The selected document must not be currently locked by another user.
3.3 Document not open by another user
The document must not be currently open by another user.

4 Post-Conditions

5 Main Flow of Events

<table>
<thead>
<tr>
<th>Situation Action/Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The Use Case begins when the Actor presses the &quot;Name&quot; button.</td>
</tr>
<tr>
<td>2. Includes <code>refresh project meta data</code></td>
</tr>
<tr>
<td>3. EMT prepares the Rename document wizard, which contains a text field for the assignment of the new name for the document.</td>
</tr>
<tr>
<td>4. The inclusion of <code>Use Case: &quot;Refresh project meta data&quot;</code> is scheduled to take place at regular intervals for as long as the Wizard is displayed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event 1</th>
<th>Event 2</th>
<th>Event 3</th>
<th>Event 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pressing the &quot;Name&quot; button</td>
<td>Refresh project meta data</td>
<td>Prepare Rename wizard</td>
<td>Scheduled at regular intervals</td>
</tr>
</tbody>
</table>