Content

1 Introduction

Many artifacts in a software development project such as requirements, system models or project management artifacts like tasks are the result of a collaborative activity and need to be accessed in a distributed setup. Furthermore most artifacts are required to be under configuration management and must therefore be in a version control system. Therefore most of the documents in a software development project are stored electronically.

Some documents such as software project management plan are text-driven and of sequential nature. For the text-driven documents we suggest to use a word processor in combination with traditional versioning systems and discussion boards. Other artifacts are model-based, such as requirements captured in a use case model or an analysis model captured in UML class diagrams. Other examples are lists of tasks assigned to project participants and contained, i.e. planned in certain iterations.

These models cannot be effectively captured in text-driven documents. This is especially true since there are interdependencies between different models, e.g. functional requirements are related to use cases. Capturing these relations often referred to as traceability is beneficial for a project and was shown to reduce the risk of project failure. All of these non-text-driven documents are of a model-based nature and therefore require an adequate representation and storage.

UNICASE is a unified CASE tool that allows capturing software engineering models in central repository. These models range from requirements, use case and UML models to the work breakdown structure and organizational models. In UNICASE all of these models can be managed and can be connected by traceability links. For example a task can be linked to the functional requirement the task is implementing. Furthermore UNICASE provides collaborative editing of all of these models and puts all of them under version control in a central repository.

However it is necessary for some of the models to be represented in a document from time to time. For example a Requirements Analysis Document needs to be presented to the client at the Analysis Review. These documents can be generated from the existing models in UNICASE and do not need to be created manually. For this purpose UNICASE offers a model-based document export facility.

In compare to other existing solution for electronically managing models and documents of a software development project, UNICASE UNI-fies the different models that otherwise reside in different tools. System Models from Requirements Elicitation, Analysis, System and Object Design as well as Project Models about Iteration Planning, Organizational Structure and Rational Management are within ONE tool. This reduces the "tool zoo" resulting in faster adoption by the students and easier access for instructors and project managers. Furthermore the tool is open source and available for free. It can be easily adapted to different methodologies or modeling techniques.

In this document we will first discuss how to get started with client and server. Then we will present how to model with UNICASE - overall, in the system model and in the project model. Finally we will give advice on how to organize documents and provide templates for documents.

1.1 Getting started

The UNICASE is a CASE-Tool integrating models from the different development activities, such as requirements, use cases, UML models, schedules, bug report and feature models into a unified model. The UNICASE client allows viewing and modifying these models in a textual, tabular and diagram visualization. The models are stored and versioned on a server called EMFStore comparable to SVN, but customized for models. UNICASE is based on the Eclipse platform including EMF and GMF. The project is open-source and released under the Eclipse Public License v 1.0 (EPL). To get started with the UNICASE Client you need to install it first, instructions can be found on the previous section or on the UNICASE website at http://unicase.org.

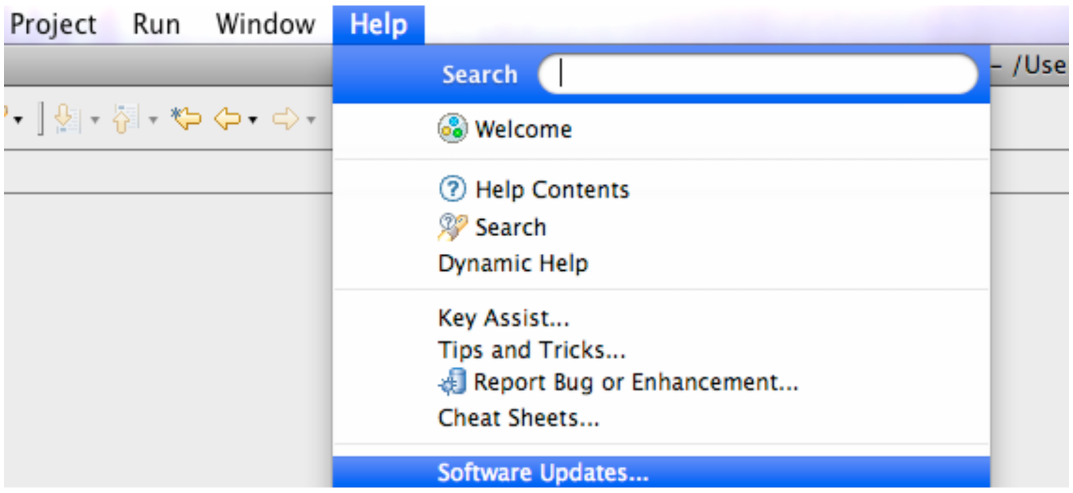
In the following we will give detailed step-by-step instructions on how install UNICASE.

We recommend you to download the "Eclipse Modeling Tools" Edition from the Eclipse download site: http://www.eclipse.org/downloads/, since it already contains all plug-ins UNICASE requires. After that you can then install the UNICASE client using the update site: http://wwwbruegge.in.tum.de/static/unicase/

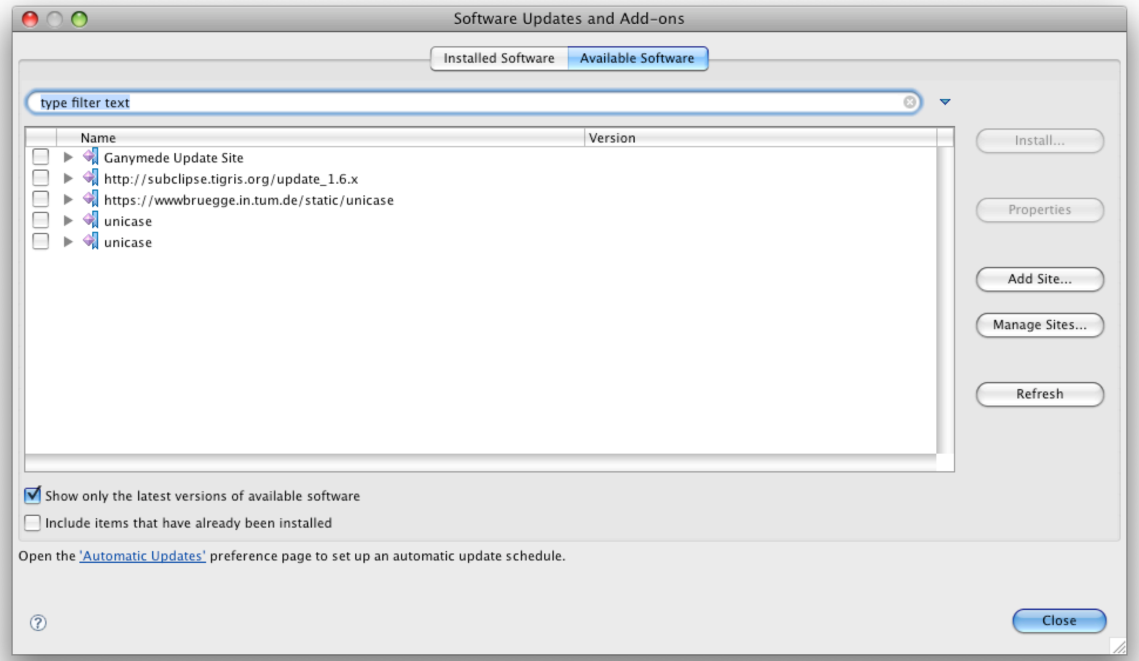
Alternatively you can download a preconfigured version of UNICASE or a standalone UNICASE client from the UNICASE homepage: http://unicase.org.

After downloading Eclipse you can then install the UNICASE client using any of the above given options by following these steps

1. Begin the installation from the Eclipse Help menu item.



2. Select "Available Software" and click on "Add Site"

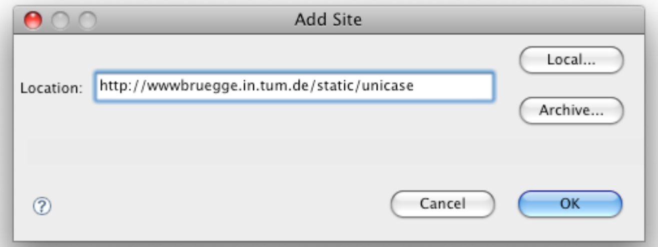


Ganymede Update Site
http://subclipse.tigris.org/update_1.6.x
https://wwwbruegge.in.tum.de/static/unicase
unicase
unicase

[✓] Show only the latest versions of available software
[ ] Include items that have already been installed

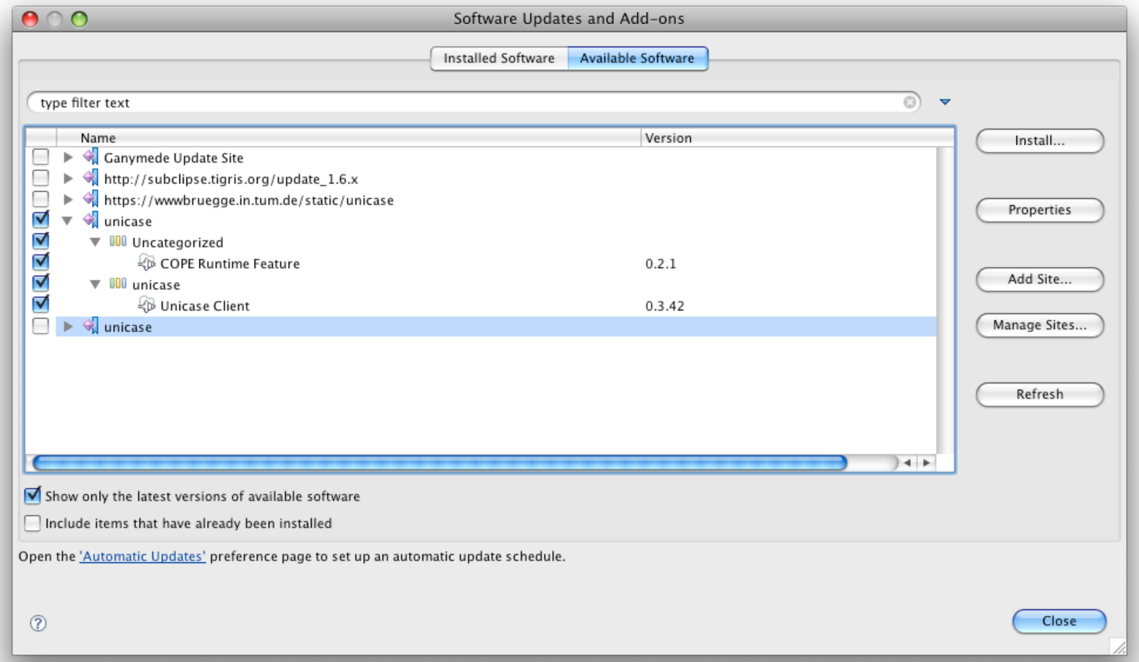
Open the 'Automatic Updates' preference page to set up an automatic update schedule.

3. Enter the URL for the UNICASE Update Site:
   http://wwwbruegge.in.tum.de/static/unicase/



Location: http://wwwbruegge.in.tum.de/static/unicase

4. Select all features from the UNICASE update site and click install. IMPORTANT: Do also add the COPE feature from the UNICASE repository.



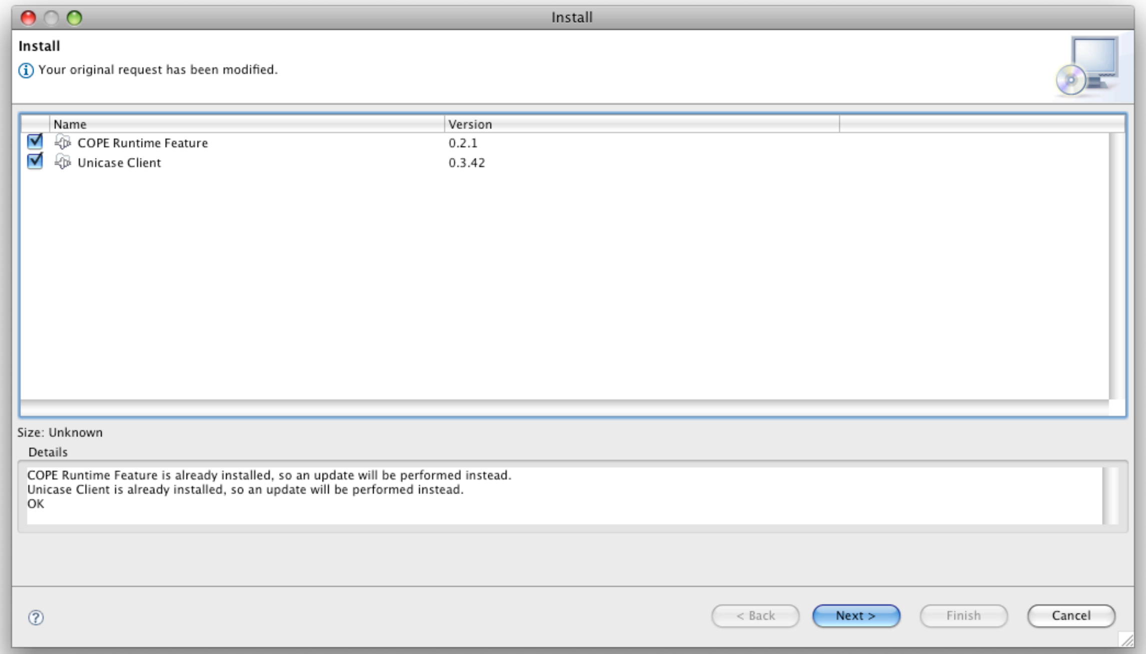
	Name	Version
☐	Ganymede Update Site	
☐	http://subclipse.tigris.org/update_1.6.x	
☐	https://wwwbruegge.in.tum.de/static/unicase	
☑	unicase	
☑	Uncategorized	
☑	COPE Runtime Feature	0.2.1
☑	unicase	
☑	Unicase Client	0.3.42
☐	unicase	

☑ Show only the latest versions of available software

☐ Include items that have already been installed

Open the 'Automatic Updates' preference page to set up an automatic update schedule.

5. Select finish.



Your original request has been modified.

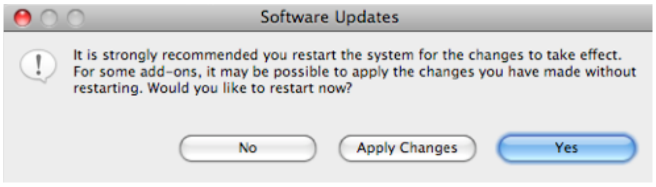
	Name	Version
☑	COPE Runtime Feature	0.2.1
☑	Unicase Client	0.3.42

Size: Unknown

Details

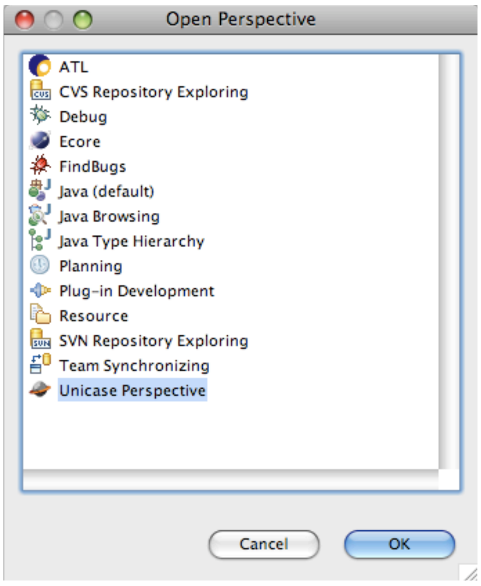
```
COPE Runtime Feature is already installed, so an update will be performed instead.
Unicase Client is already installed, so an update will be performed instead.
OK
```

6. Restart your workspace.


Software Updates

It is strongly recommended you restart the system for the changes to take effect. For some add-ons, it may be possible to apply the changes you have made without restarting. Would you like to restart now?

7. You can now open the UNICASE perspective. (Window-> Open perspective -> Other...)



1.2 First steps with the client

The UNICASE Client consists of several views to access and navigate projects. In the following we will briefly explain the most important views: "EMFStore Browser", "UNICASE Navigator" and "Model Element Editor".

1.2.1 EMFStore Browser

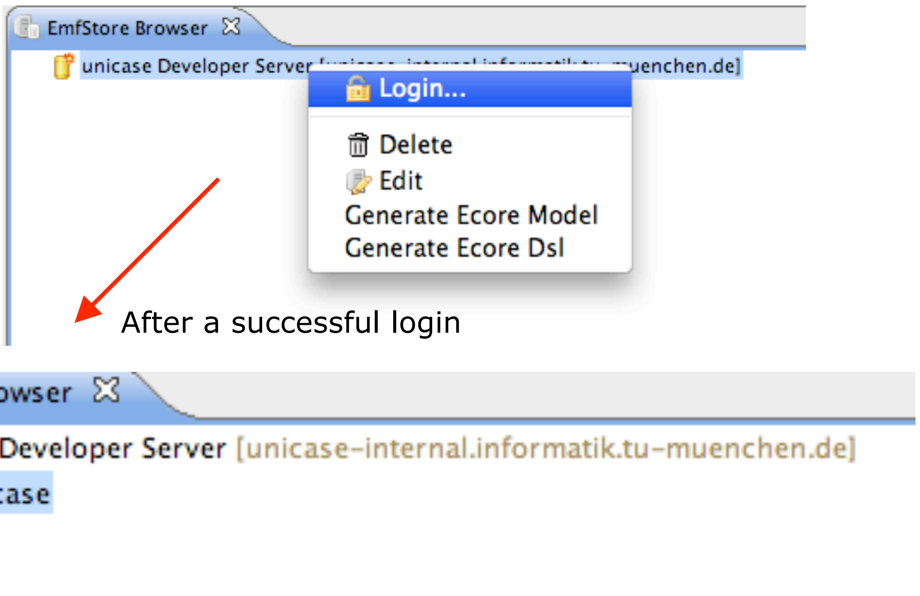
The EMFStore Browser view allows the user to access different project models from the "EMFStore" server that is a versioning system like SVN, but customized for models. Through the EMFStore Browser view the user can login to the Server and check out projects stored on the server, i.e. transferring the project from the server to the user's workspace.

Login:
A user must first be authorized to access the repository before the user checks out a project form the server. In order for the user to login the user must follow these few steps:

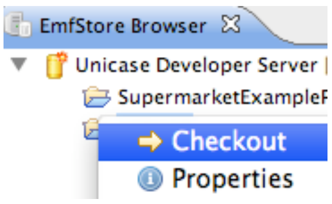
1. Open the EMFstore Browser view by selecting Window from the main menu of Eclipse, then select Show View->Other->UNICASE->EMFStore Browser.

2. Then, right click on the desired repository and select Login.

3. A window will appear requesting user login information. If the login is successful the projects belonging to that specific user will appear on the EMFStore Browser view.


EmfStore Browser
unicase Developer Server [unicase-internal.informatik.tu-muenchen.de]
Login...
Delete
Edit
Generate Ecore Model
Generate Ecore Dsl
After a successful login
EmfStore Browser
unicase Developer Server [unicase-internal.informatik.tu-muenchen.de]
unicase

Checkout:
Through the checkout the user will be able to transfer the project's content from the server to his/her local workspace. Select the desired project from the repository and then right click and select checkout. Then, the project folder will show up in the UNICASE navigator view.



Add a New Repository:
Through the EMFStore Browser view you can add a new repository to the repositories list available on your workspace. This can be done following the steps:

1. Right click away from the repository list and select new repository or simply press the button on the far right of the window of the EMFStore Browser view .

2. A new Window will appear requesting the server details.

1.2.2 UNICASE Navigator View

The UNICASE Navigator view allows the user to add, edit, view, update, share and commit projects. The Model Element Editor (MEEditor) is associated with the use of UNICASE Navigator, which allows the user to view and edit model element attributes and links between model elements.

The following section will give a quick overview on how a user can commit, update and edit a project using UNICASE Navigator

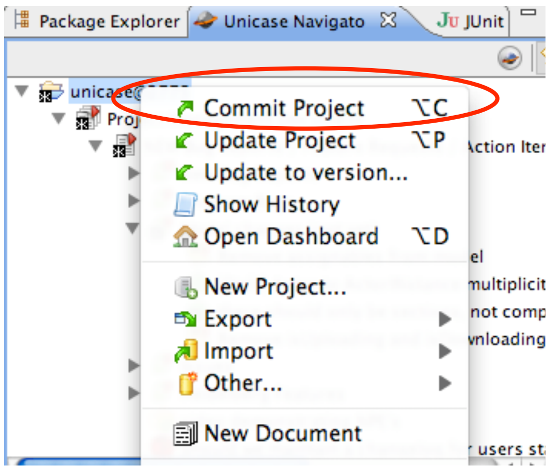
Commit a Project:
To commit a project is to share the changes that you performed on your workspace to the server, i.e. sharing it with other users. Once you perform any changes in your project a star sign will appear on your project folder and document.

This star is used to indicate that a change has been made, thus urging the user to commit the project.

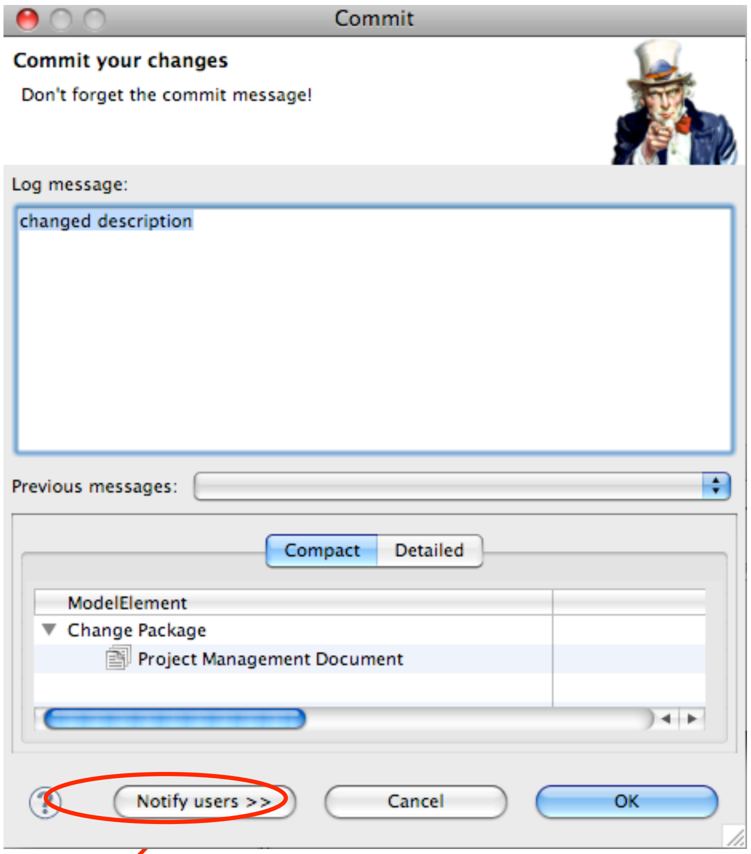
The following steps describe how to commit a project:

1. After you logged into your repository and checked out your project you have to open the UNICASE Navigator view by selecting Window from the main menu, then select Show View->Other->UNICASE->UNICASE Navigator.

2. After editing your project, you then need to select the project you just edited and then press right click select Commit Project.



3. A new window will appear requesting a log message that is used to indicate the type of changes done by the user. Through this window you can also send a notification to certain user concerning the changes you made, which will appear as a message on the dashboard. This is can be done by pressing on notify users button, another window will appear. Press the + button to add a new notification.



Commit

Commit your changes

Don't forget the commit message!

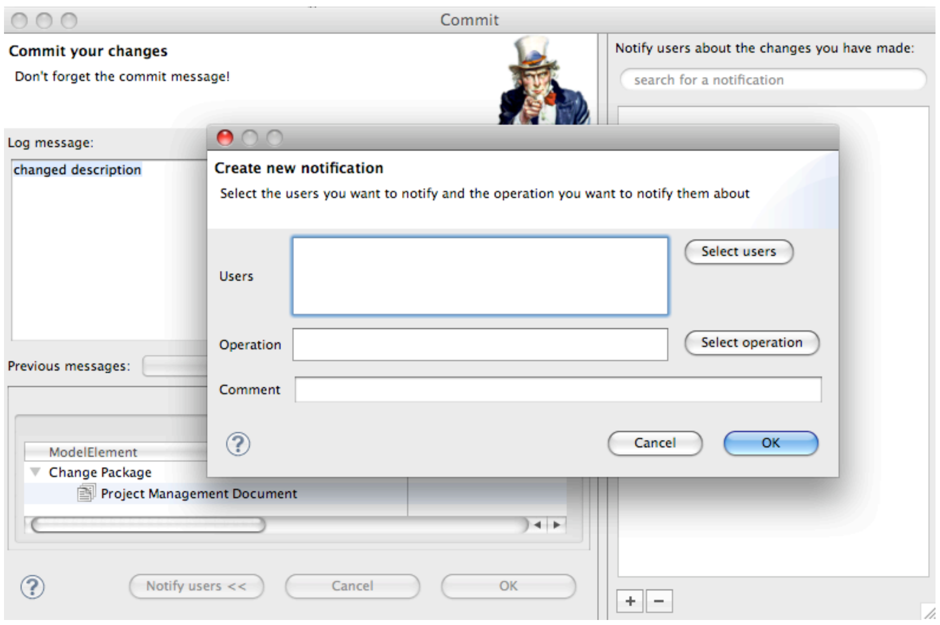
Log message:

changed description

Previous messages:

ModelElement
Change Package
Project Management Document

Used to Send Notification for users about project changes



Commit

Commit your changes

Don't forget the commit message!

Log message:

changed description

Previous messages:

ModelElement
Change Package
Project Management Document

Notify users about the changes you have made:

search for a notification

Create new notification

Select the users you want to notify and the operation you want to notify them about

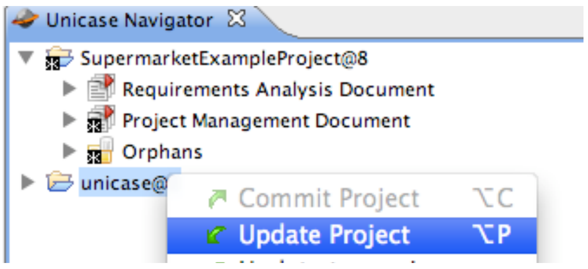
Users

Operation

Comment

Update a Project:
To update a project is to apply the changes that were performed by other users to your local workspace.

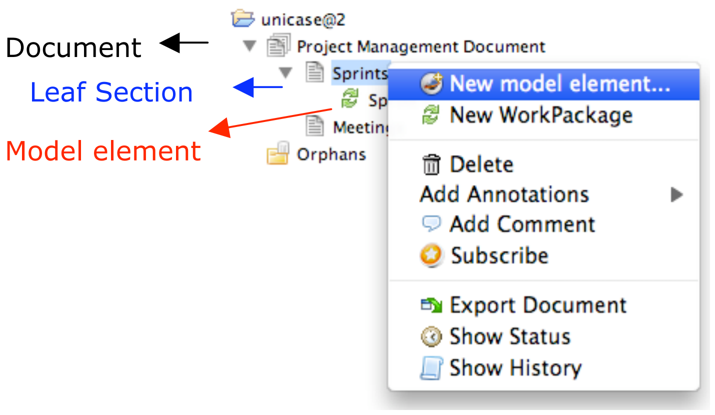
Select the project you want to update and then press right click select Update Project.


Unicase Navigator
SupermarketExampleProject@8
Requirements Analysis Document
Project Management Document
Orphans
unicase@
Commit Project ⌥C
Update Project ⌥P

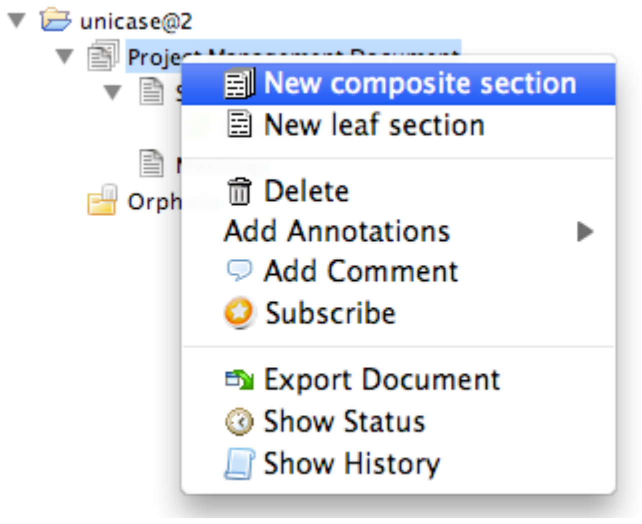
Add a new Model Element
To add a new model element to your project such as a new functional requirement, diagram, use case and scenario you need to follow these few steps:

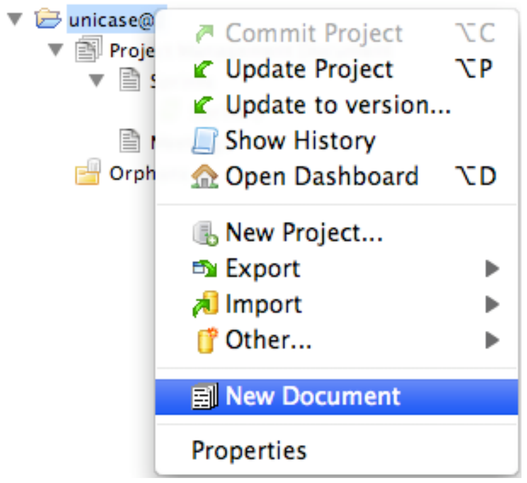
Any project is organized in a hierarchical structure where a project is divided into a group of documents and each document contains a group of composite sections and leaf sections. However, a composite section may contain other composite sections or leaf sections. Furthermore, a new model element is added only to a leaf section. To add the model element select the leaf section you would like to add the element to, press right click and then select New model element. Next a window will appear requesting the type of model element you like to add.

In case there are already model elements contained in a leaf sections, short cuts to create model elements of the existing types will be shown in the context menu of the leaf section.



Add Documents and Sections:
Documents and Sections are used to structure your project in a hierarchical way like having folders in file system. To add a new document, select the project you want to add the section to and then press right click select New Document. To add sections right click the document you just added and select New composite section or leaf section.



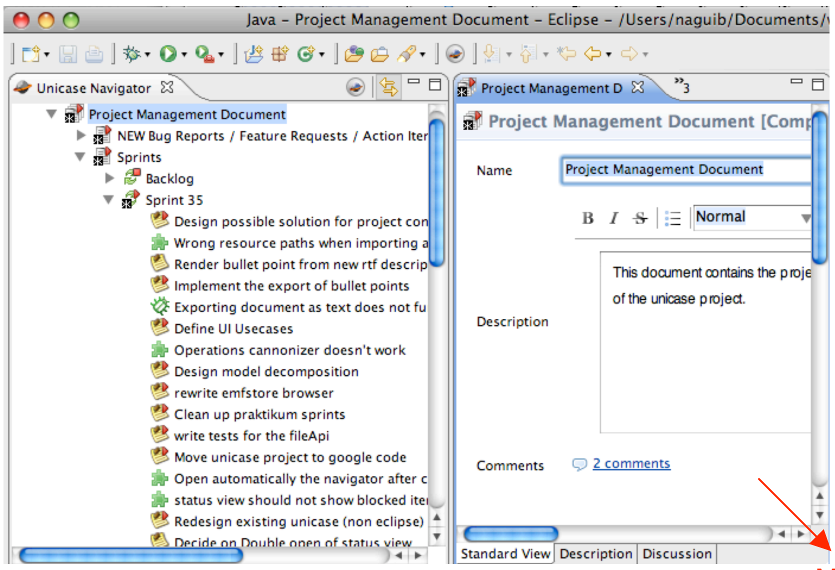


1.2.3 Model Element Editor (MEEditor)

The MEEditor allows the user to view and edit model element's attributes and links. In the following section we will give a brief outline on how to use the ME Editor.

View and Edit Model Element's Attributes:
1. Select the model element you would like to view in the UNICASE Navigator and double click on it.

2. The MEEditor window then will appear containing all the model element's attributes. Attributes can be modified in a form-based editor and will be persisted instantly upon change. The description and comment attributes are available as additional tabs also.


Java – Project Management Document – Eclipse – /Users/naguib/Documents/
Unicase Navigator
Project Management D
Standard View
Description
Discussion

- Project Management Document
  - NEW Bug Reports / Feature Requests / Action Iter
  - Sprints
    - Backlog
    - Sprint 35
      - Design possible solution for project con
      - Wrong resource paths when importing a
      - Render bullet point from new rtf descrip
      - Implement the export of bullet points
      - Exporting document as text does not fu
      - Define UI Usecases
      - Operations cannonizer doesn't work
      - Design model decomposition
      - rewrite emfstore browser
      - Clean up praktikum sprints
      - write tests for the fileApi
      - Move unicase project to google code
      - Open automatically the navigator after c
      - status view should not show blocked ite
      - Redesign existing unicase (non eclipse)
      - Decide on Double open of status view

Project Management Document [Comp

Name: Project Management Document

Normal

Description: This document contains the proje of the unicase project.

Comments: 2 comments

MEeditor

<u>Edit Links:</u>

1. Select the model element you want to add a link to it and double click on it.

2. Then, the MEEditor window will appear containing all the attributes of the model element.

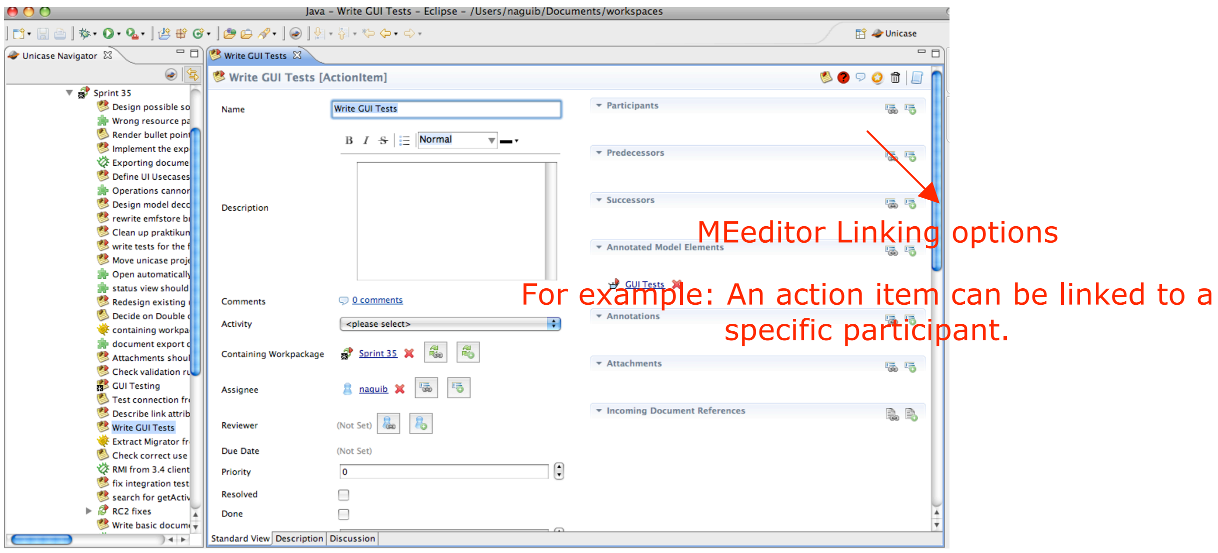
3. In the editor you will find different types of links that you can connect the model element with, for example an action item can be linked with specific participate. To add the link you will use the linking button .

4. A window will appear indicating a list of elements that can be linked to the selected model element

There are two ways to link and model element with another:

1. The first one is through the link button where you can link the model element with existing model elements that have been added previously.

2. The second one is through the create and link button where you can link an existing model element with a new model element that you need to create it first from scratch.


Java – Write GUI Tests – Eclipse – /Users/naguib/Documents/workspaces

Unicase Navigator

Sprint 35: Design possible so, Wrong resource pa, Render bullet point, Implement the exp, Exporting docume, Define UI Usecases, Operations cannot, Design model deco, rewrite emfstore b, Clean up praktikum, write tests for the f, Move unicase proje, Open automatically, status view should, Redesign existing, Decide on Double c, containing workpa, document export c, Attachments shoul, Check validation ru, GUI Testing, Test connection fr, Describe link attrib, Write GUI Tests, Extract Migrator fr, Check correct use, RMI from 3.4 client, fix integration test, search for getActiv, RC2 fixes, Write basic docum

Write GUI Tests [ActionItem]

Name: Write GUI Tests

Description:

Comments: 0 comments

Activity: <please select>

Containing Workpackage: Sprint 35

Assignee: naguib

Reviewer: (Not Set)

Due Date: (Not Set)

Priority: 0

Resolved: ☐

Done: ☐

Participants

Predecessors

Successors

Annotated Model Elements: GUI Tests

Annotations

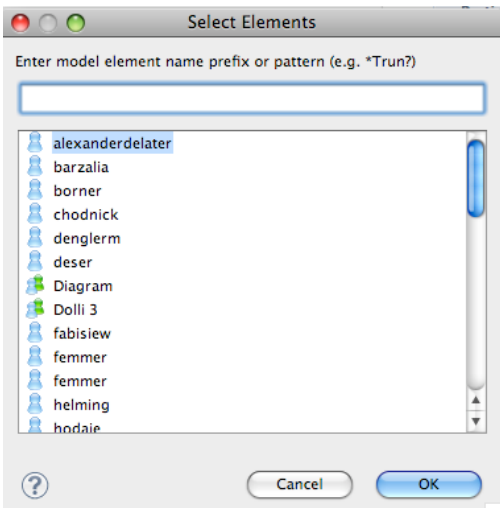
Attachments

Incoming Document References

Standard View | Description | Discussion

MEeditor Linking options

For example: An action item can be linked to a specific participant.


Select Elements

Enter model element name prefix or pattern (e.g. *Trun?)

alexanderdelater
barzalia
borner
chodnick
denglerm
deser
Diagram
Dolli 3
fabisiew
femmer
femmer
helming
hodaie

<u>Important Note:</u>
All the changes made through the MEEditor are immediately saved, but in order for other users to view the modifications made by you, the project should be committed to update the version on the server.

1.3 Setting up a server

In this section we describe how you can setup a UNICASE server. The server is called EMFStore and is responsible for storing the models in a central repository, for versioning and for access control.

Before you think about setting up a server yourself you might want to consider letting us (unicase.org) host your server. Please contact us to ask for details, our contact data is on http://unicase.org in "Who is UNICASE?".

To run the EMFStore you need Java 1.5 or higher.

Download the EMF store product from the UNICASE website and extract it. Once you have extracted it you will get a folder structure, which is dependent on the operating system you are using.

- On Windows operating system you will get the folder structure with a file named emfstore.exe.
- On Mac operating system you will get the folder structure with a file named emfstore.app (MAC application package file)
- On linux operating system you will get the folder structure with a file named emfstore.

These are the main files to start the server. You can either start the server from Graphical user interface (GUI) or console.

Start the server with the console parameter, as it will put the status message on the console. From the console you can start the server in the following way:

- Windows: emfstore.exe –console
- Macintosh: cd emfstore.app/Contents/MacOS/;./emfstore –console
- Linux:./emfstore –console

When you start the server for very first time it checks for various configuration files. Initially it will look for the emfstore configuration file i.e. "es.properties". From the status messages in console you can easily identify the location it's looking for the configuration file. By default it looks for this file in the user home directory with a folder named ".unicase". For example on Mac operating system this path can be:

- USER_HOME/ .unicase/emfstore/conf/es.properties

This configuration file can be used to setup a number of parameters ranging from accepted versions of the clients to various authentication policies (ldap or file based) and the interval to take the backup of server state. Authentication details for the super user (administrator) can also be given in this file. Another way of authorizing the users (incase if you don't have ldap server) is a plain file with unix like authentication details. This file is in the same directory and is named "user.properties" file. A sample of the "es.properties" file is given below:

```
# es.properties
# properties are described in ServerConfiguration.java
emfstore.acceptedversions=
emfstore.accesscontrol.authentication.superuser =
emfstore.accesscontrol.authentication.superuser.password =
emfstore.persistence.version.projectstate.everyxversions =
emfstore.persistence.version.backup.projectstate.everyxversions =


emfstore.validation =
emfstore.validation.level =
emfstore.validation.exclude =

# new certificate for beta and main server
emfstore.keystore.password = lFhjgasdjfaw4gtawe4gtkasdf
emfstore.keystore.alias = unicaseserver

emfstore.accesscontrol.authentication.policy=ldap
emfstore.accesscontrol.authentication.ldap.1.url=
emfstore.accesscontrol.authentication.ldap.1.base=
emfstore.accesscontrol.authentication.ldap.1.searchdn=
```

Thus the very first time if EMFSTORE server doesn't get's any of the previously mentioned configuration file, it will show a warning on the console. Create these configuration files and to restart the server. Do not forget to mention the super user's "name" and "password" (you are free to choose any username and password combination) that you can use for managing the server from the client.

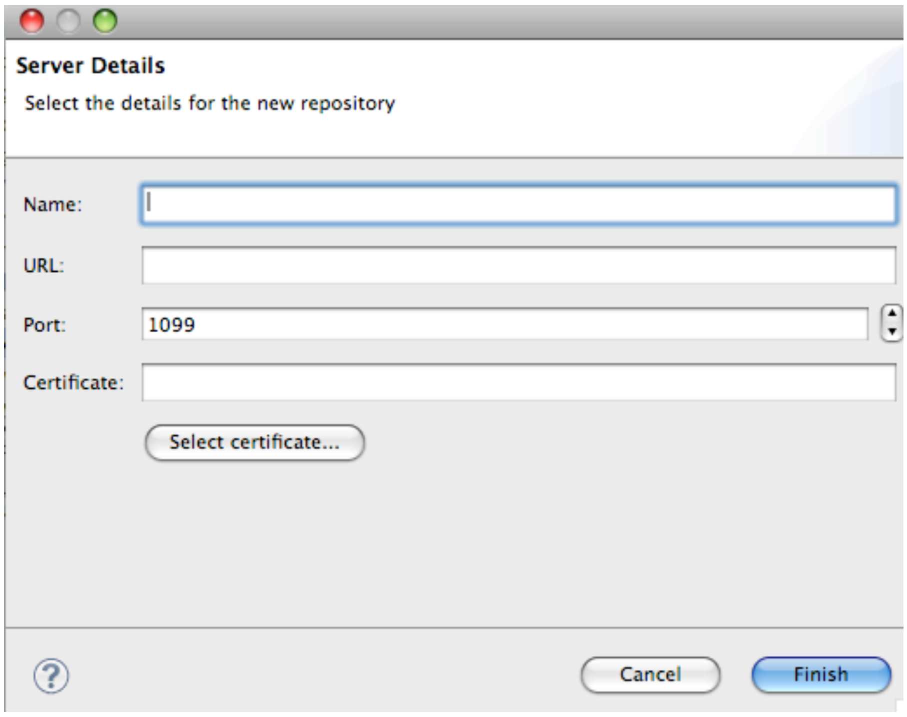
<u>Connecting to the server:</u>
You can follow the general guidelines to install the UNICASE client on the eclipse platform. Once you have installed the client restart your eclipse instance and open the UNICASE perspective. In the following perspective you can see the "EMFStore Browser" at the bottom panel where you can add the newly created repository (EMFStore Server) by following the steps given below.

<u>Add a New Repository:</u>
Through the EMFStore Browser view you can add new repository to the repositories list available on your workspace. This can be done following the steps:

1. Open the EMFStore Browser view.

2. Right click away from the repository list and select new repository or you can simply press the button on the far right of the window of the EMFStore Browser view .

3. A new Window will appear requesting the server details.


Server Details

Select the details for the new repository

Name:

URL:

Port: 1099

Certificate:

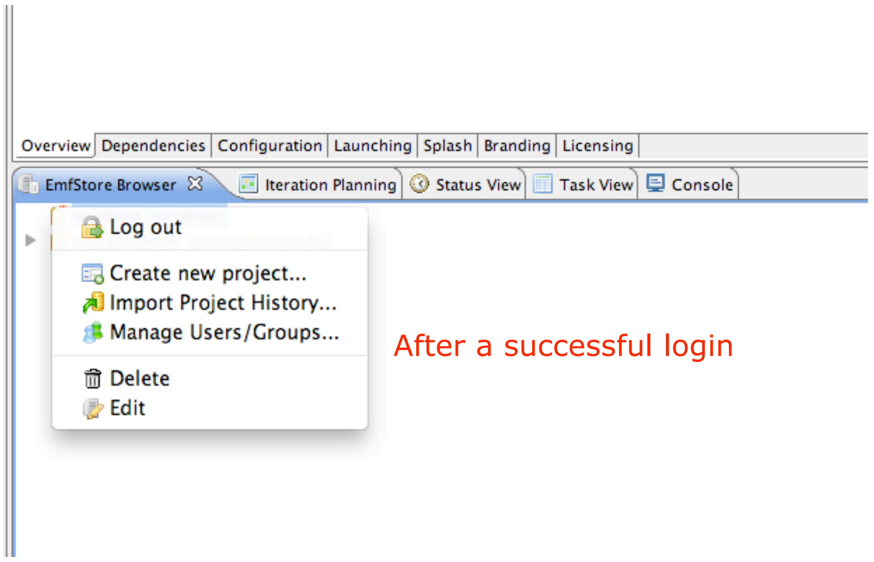
4. Insert a name for your sever to identify it easily in the Emfstore browser list.
5. Insert the host-name for the server or IP address, in case if the client and server is on the same machine then insert “localhost”.
6. By default port is 1099.
7. Click on the select certificate tab and choose “unicase.org 2009#1”. You can also have your own certificate.
8. Click on finish.

Now you will be able to see your server listed in the EMFStore Browser. The next step is to login in the server.

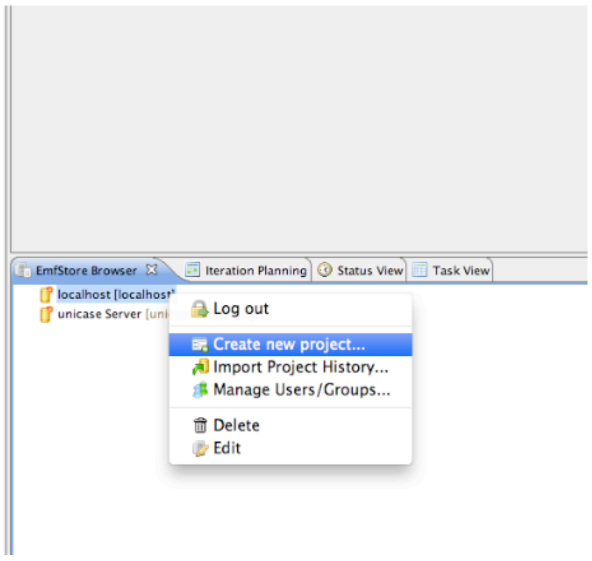
The Administrator must be authenticate first to access the Repository. In order to follow these few steps:

1. Open the EMFStore Browser view by selecting Window from the main menu of Eclipse, then select Show View->Other->UNICASE->EMFStore Browser.
2. Then, right click on the newly created repository and select Login.
3. A window will appear requesting user login information, type the super user’s “name” and “password” in their respective fields and click OK to authenticate.
4. If the login detail is correct EMFStore Server will check for the client version (current) and the accepted versions in the “es.properties” file to be the same. If same then it will login else reject the login request telling “Client version not compatible with server. Please update your client”.

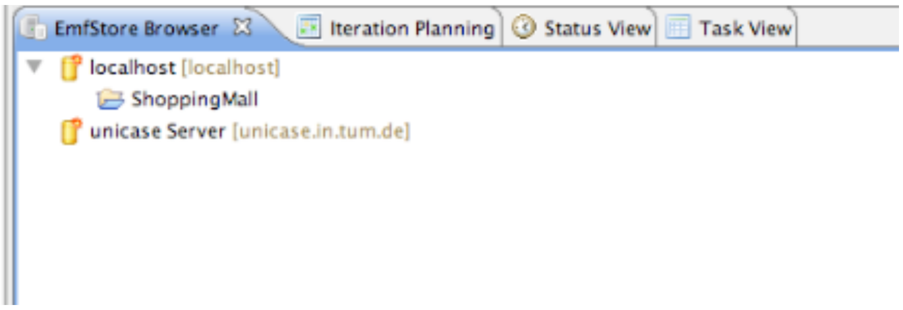
Once successfully logged in, you can right click on the repository node and perform various administration tasks like creating project, managing users and groups etc.



<u>Creating a new project:</u>
With the server administrator authorization login the user can create and manage projects.

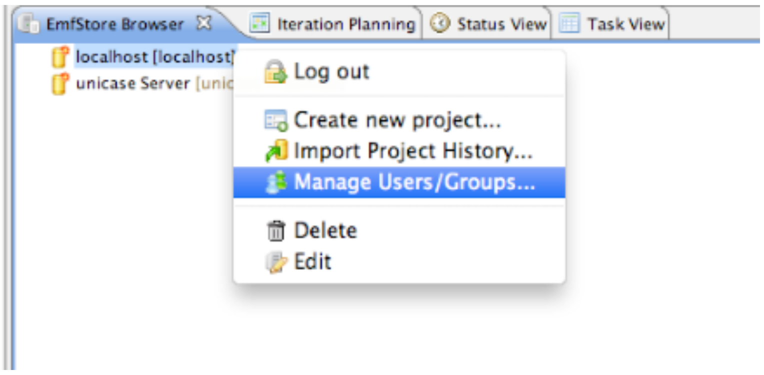


To do this just right clicks on the repository after super user login and click on " Create new project". Name the new project and insert the description at the given place, finally click on OK button.

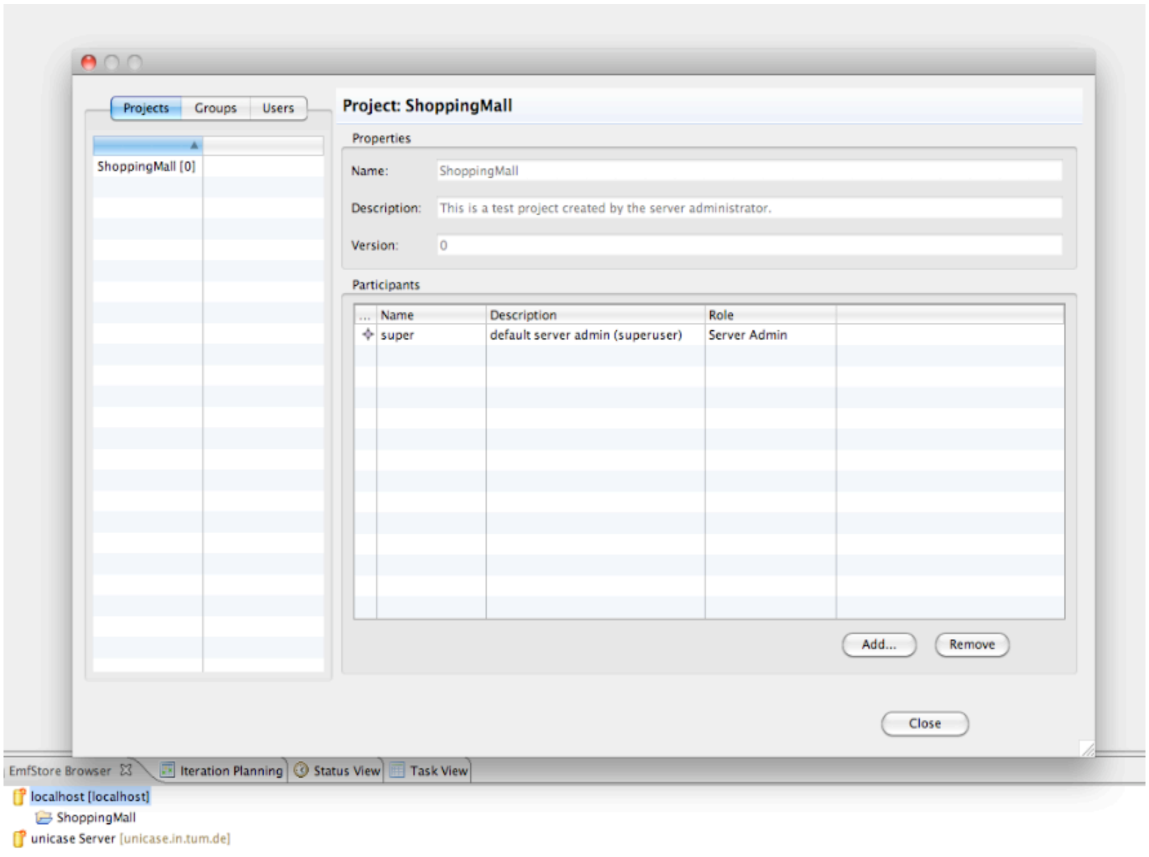


Now you can see the newly created project in the repository.

Managing Users and Groups:
Once you click on the repository you can see the tab to manage users and groups. From this tab you create users, assign them in a group, assign them one or more projects and more over you can set the permissions any particular user is having on any given project.



Once you click on the above highlighted menu item, you will see the following menu:


Projects | Groups | Users

ShoppingMall [0]

Project: ShoppingMall

Properties

Name: ShoppingMall

Description: This is a test project created by the server administrator.

Version: 0

Participants

...	Name	Description	Role
✣	super	default server admin (superuser)	Server Admin

Add... Remove

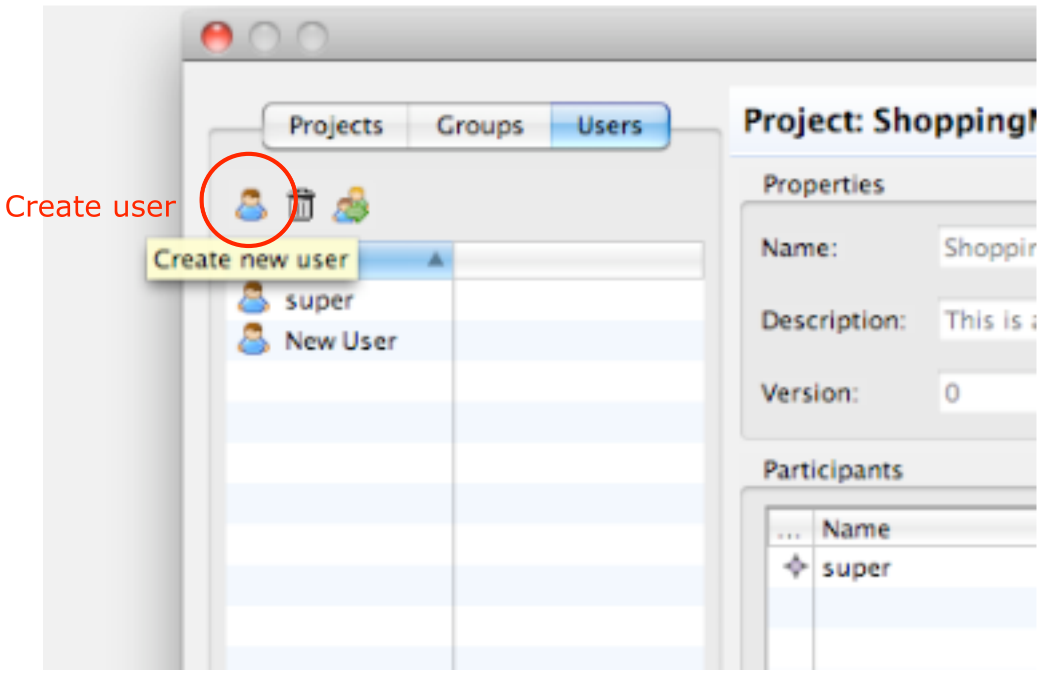
Close

EmfStore Browser | Iteration Planning | Status View | Task View

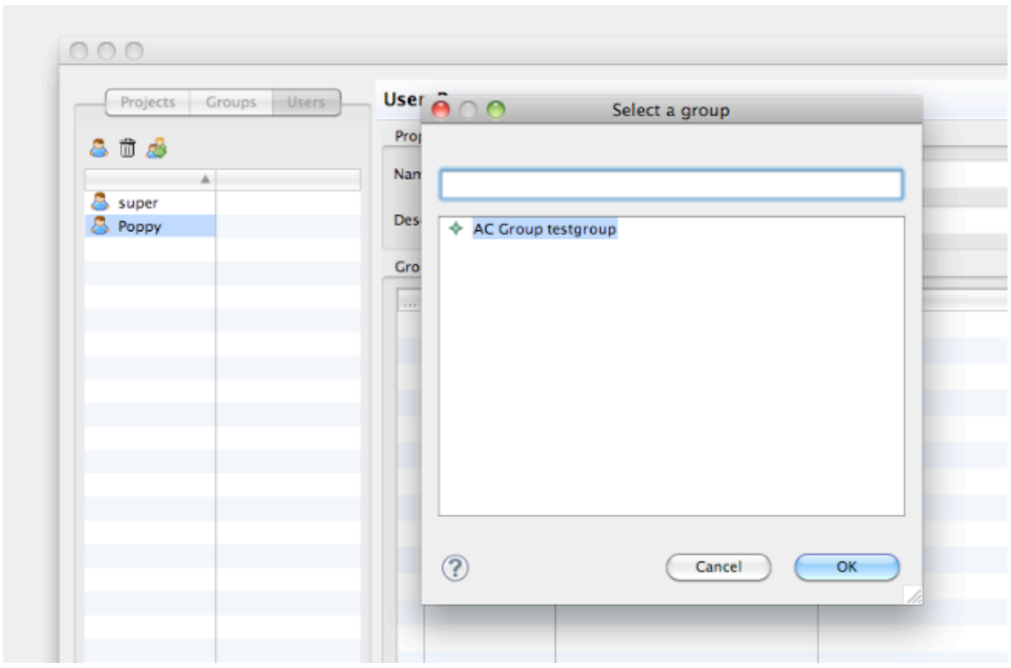
localhost [localhost]
ShoppingMall
unicase Server [unicase.in.tum.de]

On the left hand side menu you can see all the projects available in the repository. On the right hand side you can see the description of the project and finally at the bottom right it shows the Users/participants assigned to the project. Currently it is having only one participant and that is "super" user having role "Server Admin".

On the left top there are two more tabs to manage "Groups" and "Users". To add a new user click on the button shown in the figure below:

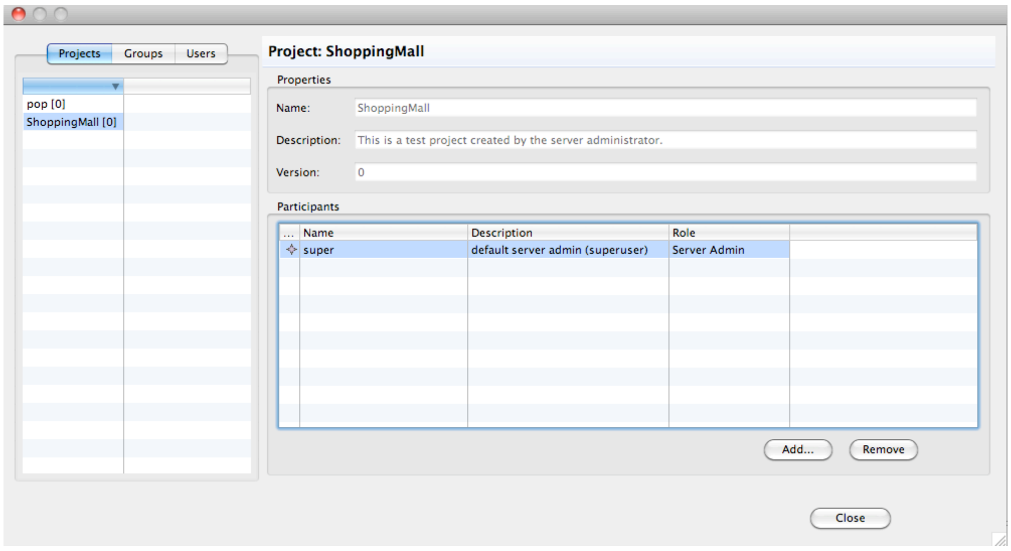


Once you click on it a user with default name "New User" is created. You can click on the newly created user and modify its details like his/her name, add him/her in a group or remove (Add and Remove tab at the right bottom) from any previously assigned group. Once you click on the "Add" button you can see all the existing groups, in our case there is only once group currently (see below). Select the desired group and hit the OK button.



To create groups click on the "Groups" tab on the top left to create one.

To make a project accessible to a group or user you can add a group or user to the participants of a project.



Project: ShoppingMall

Properties

Name: ShoppingMall

Description: This is a test project created by the server administrator.

Version: 0

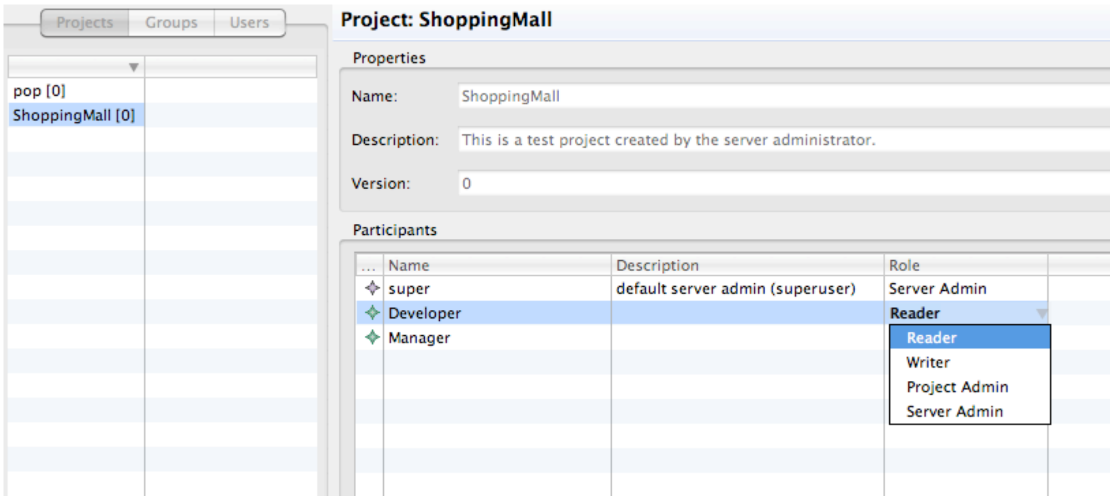
Participants

...	Name	Description	Role
	super	default server admin (superuser)	Server Admin

Once you have added the groups or users o the participants of a project you can set the role of the group or user to four different access levels:

- Reader: Only read access to the project
- Writer: Read and write access

- Project Admin: Writer with additional privilege to delete documents
- Server Admin: Project Admin for any project, may even delete projects from the server and administer users



pop [0]
ShoppingMall [0]

Project: ShoppingMall

Properties

Name: ShoppingMall

Description: This is a test project created by the server administrator.

Version: 0

Participants

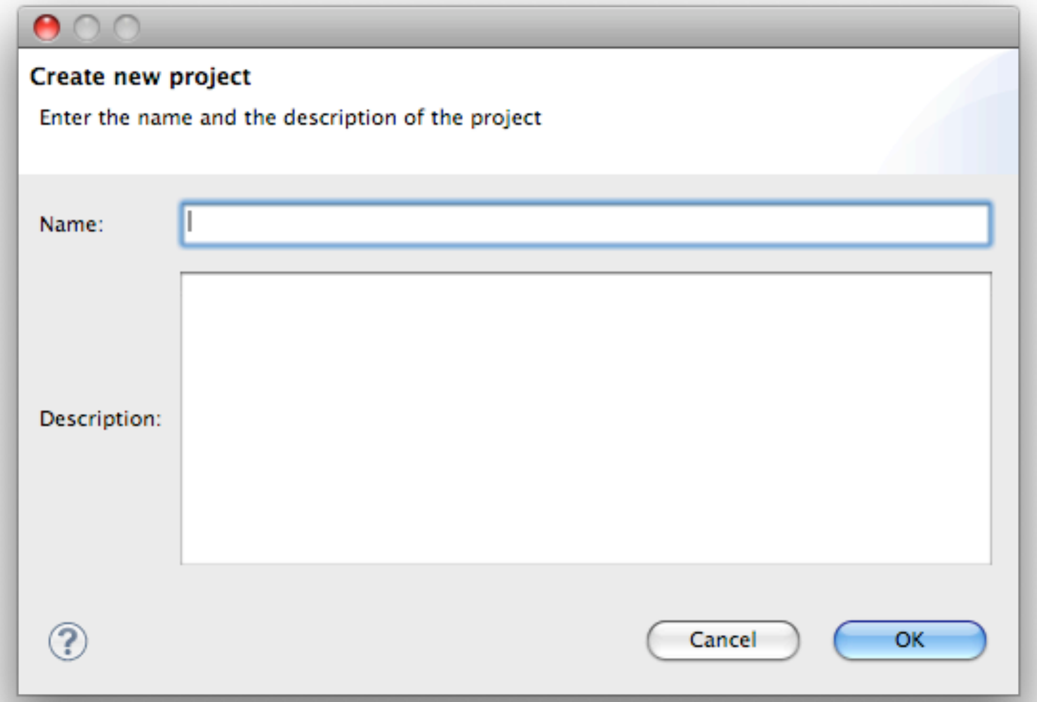
Name	Description	Role
super	default server admin (superuser)	Server Admin
Developer		Reader
Manager		

<u>Add a New Project:</u>

To add a new project into a specific repository you need to do the following:

1. Log into the UNICASE Server (EMFStore Browser)

2. Right click on the Server and choose “Create new project“. A wizard for the project creation opens.



Create new project

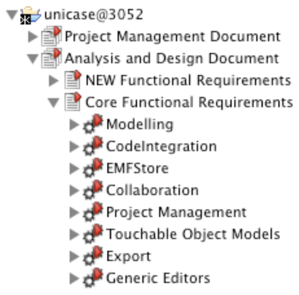
Enter the name and the description of the project

Name:

Description:

2 Modeling in UNICASE

All artifacts of a software development project are organized in a *Project* in UNICASE. The artifacts in UNICASE consist of *ModelElements* and *Links*. A model element is a node such as an action item. A Link connects two model elements, such as a link from a requirement to an action item for example.



Model elements are organized in a hierarchical document structure, the document model. A *document* is at the root level of a project and consists of sections. A *section* is either a composite or a leaf section. A *composite section* can contain leaf or composite sections. A *leaf section* contains other model elements. For example a Requirements Analysis Document may contain a composite section "Requirements" with a contained leaf section "Functional Requirements" containing all functional requirements of the project.

Every model in UNICASE is based on the concept of containment. Each model element can only have one (parent) container. In the example of the requirements analysis document, the document is contained in the project, the "Requirements" section is contained in the document and the leaf section "Function Requirements" is contained in the composite section "Requirements". Apart from links that reflect a containment relationship, there are reference links. While a model element can only have one container it can have many referenced model elements. In tree views, such as the Navigator, UNICASE displays the containment hierarchy of a project in general.

We distinguish two different types of models in UNICASE – System Models and Project Models. System Models describe the system under development and will be detailed in the following section. Project Models describe the project itself and will be described after the System Models.

2.1 System Modeling

System Models in UNICASE are all models that describe the system under development, these include:

- Scenario Model
- Tree-based Requirement Model
- Enhanced UML Use Case Model including Use Case Diagrams
- UML Class Model including Class Diagrams
- UML State Model including State Diagrams
- UML Activity Model including Activity Diagrams

- Subsystem Decomposition Model based on UML packages
- UML Component Model including Component Diagrams

We will describe the intended usage of the system models for the different Software Engineering activities in the following subsections.

2.1.1 Requirements Elicitation and Analysis

We suggest the use of *Scenarios* to describe a concrete flow of events in the system under development. A Scenario is a number of steps that describe a user interacting with the system in a concrete instance. It includes real input values and expected system outputs. In UNICASE you can create Scenarios and link them with *Actor Instances* that are initiating or participating in the scenarios.

To formalize a scenario a *Use Case* can be used. It contains an abstract flow of events describing the interaction of an actor with the system. An *Actor* is not a concrete user but a user role. Actors can be linked to the actor instances that take the actor role. A scenario can be linked with the use cases it is an instance of.

Functional Requirements can be used to model an expected behavior of the system from a functional point of view. Functional requirements can be refined to smaller but more precise refining functional requirements. Therefore the functional requirements resemble a tree-structure. Functional requirements can be linked to the use cases where they are used to realize the use case. In general one use case uses a number of requirements and one requirement can be used in several use cases. *Non-functional Requirements* complete the requirements of a system by providing quality constraints on the system under development such as performance. They can be linked to the use cases where they need to be considered and which they constrain.

Classes can be used to identify entity, boundary and control objects in the analysis model. They consist of attributes and operations. Classes can link to other classes with inheritance, aggregation, composition and association links. *Packages* are used to organize classes in a tree-structure. Classes are contained in packages.

States can describe the behavior of a system for a class. The can be linked to other states. Activities describe the behavior of system from an action-centric viewpoint. They can be linked to other activities. Both, state and activity models, are used to build the dynamic model of a system.

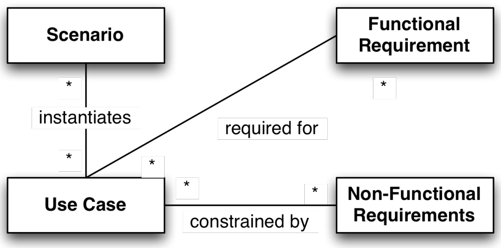


Figure 1 - Requirements Model

2.1.2 System Design

Packages can represent subsystems in the system decomposition in system design. Therefore every package can define a facade class that shows which operations of a subsystem are available to other subsystems. Façade classes are an effective way to define and discuss the functionality different subsystems offer.

Components consist of a number of services that describe the functionality a component offers. A component can be linked to other components it is using services of. Also a component can be linked to the subsystems it relies on.

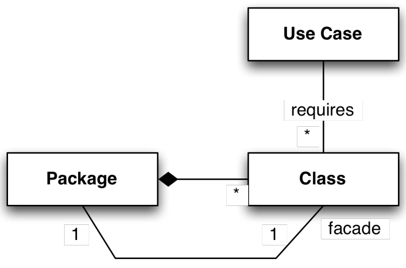


Figure 2 - Subsystem Model

2.1.3 Object Design and Code Generation

The purpose of modeling a detailed Object Model with Packages and Classes in UNICASE should be code generation, otherwise the overhead for modeling is too big and the Object Model should then be documented in the code. From an Package in UNICASE an EMF Ecore Model can be generated. This Ecore Model can be input to a variety of code generation frameworks, such as the Eclipse Modeling Framework or the Open Architecture Ware.

2.2 Project Model

The Project Model describes the Development project itself. This includes the tasks of a project, the projects rational management models and the organizational structure. The following models are part of the project model:

- Task Model
- Rationale Management Model
- Communication Model
- Organizational Model
- Defect Management Model
- Iteration Planning Model
- Meeting Model
- Release Management Model

In the following subsections we describe the intended use of each model of the project models.

2.2.1 Task Model

Tasks are called work items in UNICASE. Every work item can be assigned to exactly on user, but many participants can help in completing a task. The assignee of a work item is he user who is responsible for its completion. Since multiple responsible people for a work item of lead to confusion and delay in task completion we explicitly only allow one assignee. You can use the participant attribute to define more users contributing to a task. A reviewer can be assigner to work item to indicate that this user should review the results of a work item after its completion. A work item can be in the resolved state (resolved is checked); this means that the work is completed but waiting for review.

Work items can be annotated to other model elements indicating that they are concerned with performing something in relation to or directly with the annotated elements. This helps to find additional information for a work item. For example annotation an implementation task to the corresponding functional requirement helps the developer in finding out details about the expected behavior of the system.

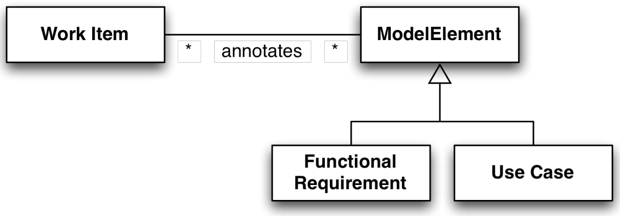


Figure 3 - Annotations

There are four different subtypes of work items: ActionItem, Issue, BugReport and WorkPackage.

ActionItems are small units of work that can be coped with by a developer in a restricted amount of time. Action item can be set to done if they have been resolved and reviewed.

Issues are problems that are not yet decided on how to resolve. The assignee of an Issue helps to find a solution for an issue but does not implement this solution. Once a solution is found an action item can be created to implement the solution if necessary.

BugReports describe a system failure that occurred and how the underlying fault was fixed. We will detail the Defect Management Model in a subsequent section. Bug reports can be set to done if they have been resolved and reviewed.

Finally *WorkPackages* consist of a number of work items and represent an aggregation of work to be done. They can be used to model a work breakdown structure by applying the composite pattern. WorkPackages help to organize work items to reduce complexity.

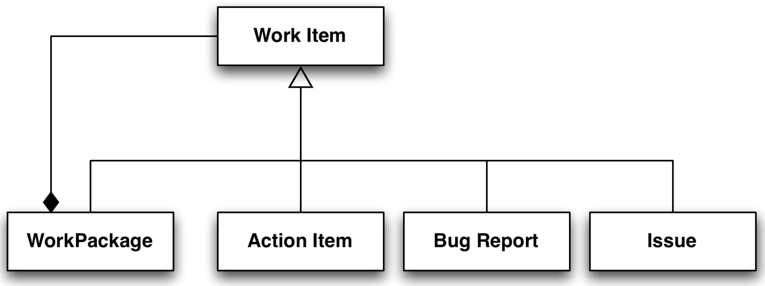


Figure 4 - Work Item Model

2.2.2 Rationale Management Model

The purpose of Rationale Management is to preserve the rationale of decisions and help in making informed decisions.

Issues are the primary model element for rationale management. Issues represent an open question or problem that needs to be resolved. Issues contain Proposals.

A *Proposal* represents a possible solution to an Issue. Issues also may contain a Solution. Each proposal can be rated along a number of Criteria.

A *Criterion* is an important influence factor on the decision represented by the issue. Criteria are often non-functional requirements.

Finally a *Solution* solves an Issue and references all proposals it is based on. An issue, which contains a solution, is considered solved. To reopen an issue you can remove or delete its solution.

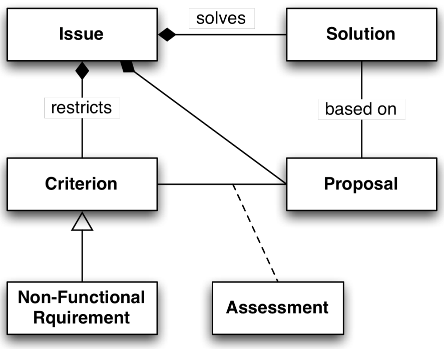


Figure 5 - Rationale Model

2.2.3 Communication Model

Comments can be added to any model element in UNICASE. Comments can be replied on. Thereby a discussion can be facilitated within UNICASE. Discussing within UNICASE in the context of a model element has the advantage of a direct traceability to the objects under discussion.

2.2.4 Organizational Model

The organization of a development project can be model by Users and Groups.

A *User* represents a human being in the project such as a developer, manager, client, stakeholder and others. Users can be organized in Groups.

A *Group* refers to a number of member users or groups. Groups can be used to model development teams or functional teams (e.g. management team). A user can participate in multiple teams.

2.2.5 Defect Management Model

Defects can be recorded and tracked in the defect model. *BugReports* represent a report about a system failure or a feature request. Also it contains information about the current status of the bug resolution or feature implementation. Since a bug report is a work item it can be assigned to a user who is responsible for resolving the bug. Steps to reproduce a bug should be provided in its description.

Bugs should be annotated to the component or class they impact, as soon as this information becomes available.

2.2.6 Iteration Planning Model

For planning iterations work items allow to specify predecessor and successor work items. A predecessor work item needs to be completed before its successor work item can be completed. Work packages can be used to model iterations and all the work to be performed in the iteration. A work package (iteration) will automatically be flagged as completed as soon as all its contained work items are completed, also this will set its successor work package automatically to the ready state. The work items contained in a work package can be assigned to users and can be assigned a priority.

2.2.7 Meeting Model

To capture the content of meetings, UNICASE supports a meeting model. A meeting consists of a number of meeting sections. *MeetingSections* can be composite, work item or issue leaf sections. A *CompositeMeetingSection* contains other meeting sections. A *WorkItemMeetingSection* references work items. An *IssueMeetingSection* references issues. When a new meeting is created a default template is used to generate its sections. In the work item section in information exchange users can report on their status on the references work items. In the work items sections in wrap up users can record new work items that were identified during the meeting. In the issues meeting section Discussion users can reference issues that should be discussed during the meeting. Agendas and Protocols of the meetings can be generated with the document export to create PDF documents. A meeting is exported as an agenda if the meeting time is still in the future, otherwise it is exported as protocol.

2.2.8 Release Management Model

To model the roadmap of development project UNICASE provides Milestones. A Milestone references model elements that are intended to be completed in the respective milestone. We recommend using functional requirements or using cases in the milestones since they represent functionality the system is supposed to expose and are therefore testable.

2.3 UNICASE Views and Features

UNICASE provides a number of views to ease the management of specific models or to help in the creation of models.

- Task View: Shows the current work of a user
- Status View: Shows the status of a model element
- History View: Shows the history of a project or model element
- Dashboard: Shows current important changes in the project
- Document Export: Exports models to documents
- Code Generation: Generates Code from an Object Model
- Validation View: Validates all models in a project

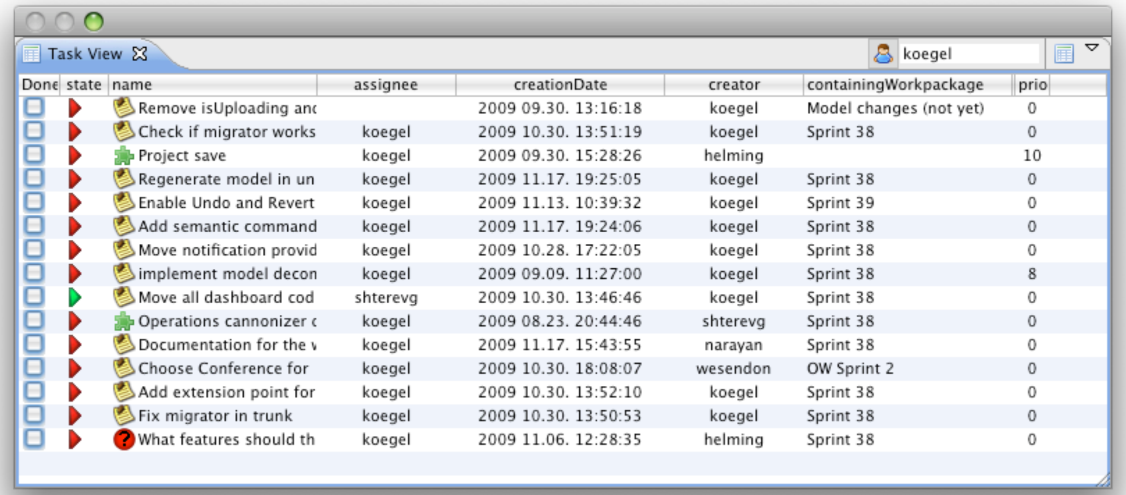
In the following sections we will detail each view and feature and explain how its intended use.

2.3.1 Task View

The task view shows the current work items of the current user of the current project. It only shows work items that the user need to resolve and work items that the user needs to review. It is part of the UNICASE perspective but can also be invoked from the menu "Window -> Show View".

The process for every work item subtype is similar. A work item is not resolved and not done at the beginning. If the item is not assigned to anyone it is shown on the task view of is creator. If it is assigned but not resolved it is shown on the assignee's and the participants´ task views. If it is resolved and a reviewer is assigned it is shown on the reviewer's task view. If it is resolved but no reviewer is assigned it stays at the assignee's and the participants´ task views. The purpose of this process is that a work item always visible on a task view of a user as long as it is not resolved.

The task view shows only the work items that are currently in the ready state. Items are in the ready state if they can be performed at the moment according to the task model. In other words they are in the ready state if their predeccessors are completed. Otherwise they are blocked and all their contained work items will not show on the task view since they cannot be performed anyway.



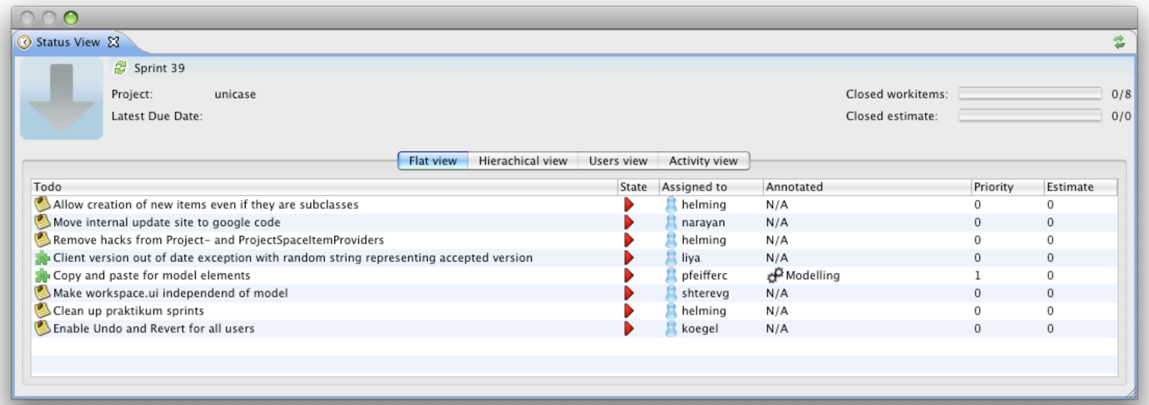
Done	state	name	assignee	creationDate	creator	containingWorkpackage	prio
☐	▶	Remove isUploading anc		2009 09.30. 13:16:18	koegel	Model changes (not yet)	0
☐	▶	Check if migrator works	koegel	2009 10.30. 13:51:19	koegel	Sprint 38	0
☐	▶	Project save	koegel	2009 09.30. 15:28:26	helming		10
☐	▶	Regenerate model in un	koegel	2009 11.17. 19:25:05	koegel	Sprint 38	0
☐	▶	Enable Undo and Revert	koegel	2009 11.13. 10:39:32	koegel	Sprint 39	0
☐	▶	Add semantic command	koegel	2009 11.17. 19:24:06	koegel	Sprint 38	0
☐	▶	Move notification provid	koegel	2009 10.28. 17:22:05	koegel	Sprint 38	0
☐	▶	implement model decon	koegel	2009 09.09. 11:27:00	koegel	Sprint 38	8
☐	▶	Move all dashboard cod	shterevg	2009 10.30. 13:46:46	koegel	Sprint 38	0
☐	▶	Operations cannonizer c	koegel	2009 08.23. 20:44:46	shterevg	Sprint 38	0
☐	▶	Documentation for the v	koegel	2009 11.17. 15:43:55	narayan	Sprint 38	0
☐	▶	Choose Conference for	koegel	2009 10.30. 18:08:07	wesendon	OW Sprint 2	0
☐	▶	Add extension point for	koegel	2009 10.30. 13:52:10	koegel	Sprint 38	0
☐	▶	Fix migrator in trunk	koegel	2009 10.30. 13:50:53	koegel	Sprint 38	0
☐	▶	What features should th	koegel	2009 11.06. 12:28:35	helming	Sprint 38	0

2.3.2 Status View

The status view shows the status of a model element. If the model element is a work package it will show all contained work items. If the model element is a system model element it will show all work items that are still unresolved to complete this element. For example if the element is a functional requirement the status view shows all work items that are still required to complete the functional requirement. The status view can be invoked from the context menu of every model element.

The status view has four different representations: flat, hierarchical, user and activity view. Flat shows all work items in a flat view while hierarchical mode relies on the containment structure to show a tree of work items and their origin. User mode shows the work items segmented by assignees. The activity mode segments the work items by the activities they are assigned to.

The status view is a good instrument to get an overview of the current status of a work package or a functional requirement. Also it can be used to plan iterations effectively. To plan an iteration open the corresponding work package in the status view. You can drag a functional requirement (or in general any system model element) onto the status view flat mode panel. All open work items of the system model element will be collected and added to the iteration. Repeat this process to add every item to be completed in the iteration. To assign users to the work items and to see the current assignment situation switch to the user mode. You can assign items by drag and drop.



Sprint 39

Project: unicase

Latest Due Date:

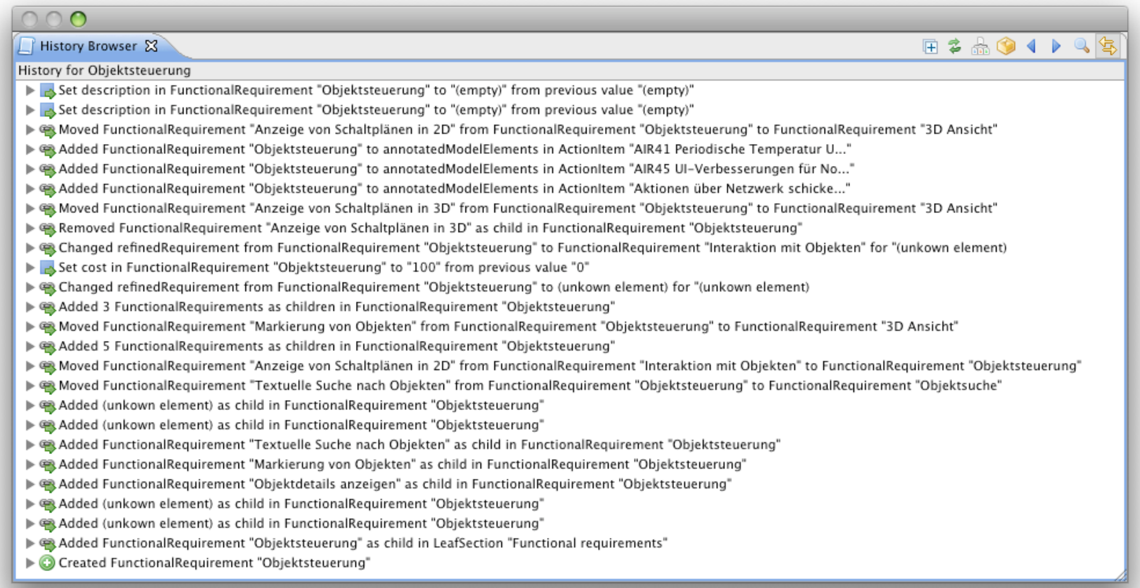
Closed workitems: 0/8

Closed estimate: 0/0

Todo	State	Assigned to	Annotated	Priority	Estimate
Allow creation of new items even if they are subclasses	▶	helming	N/A	0	0
Move internal update site to google code	▶	narayan	N/A	0	0
Remove hacks from Project- and ProjectSpaceItemProviders	▶	helming	N/A	0	0
Client version out of date exception with random string representing accepted version	▶	liya	N/A	0	0
Copy and paste for model elements	▶	pfeifferc	Modelling	1	0
Make workspace.ui independend of model	▶	shterevg	N/A	0	0
Clean up praktikum sprints	▶	helming	N/A	0	0
Enable Undo and Revert for all users	▶	koegel	N/A	0	0

2.3.3 History View

The history view shows the history of a model element or a project. You can invoke the view from the context menu of a model element or the project respectively. It will always show the newer changes at the top. When invoked on a project it shows all changes on the projects and their revision numbers. When invoked on a model element it shows the changes that have been performed on the model element recently. You can show or hide the revision numbers by pressing the yellow button. You can switch to a different representation where the changes are grouped by the elements they occurred on by pressing the hierarchy button. If you expand a change it shows the model elements that were impacted by the change. On mouse over author revision and date of each change can be displayed. The history browser can be linked with the navigator by the arrow button. The navigator will then always highlight the element of the change that is currently selected in the history view.

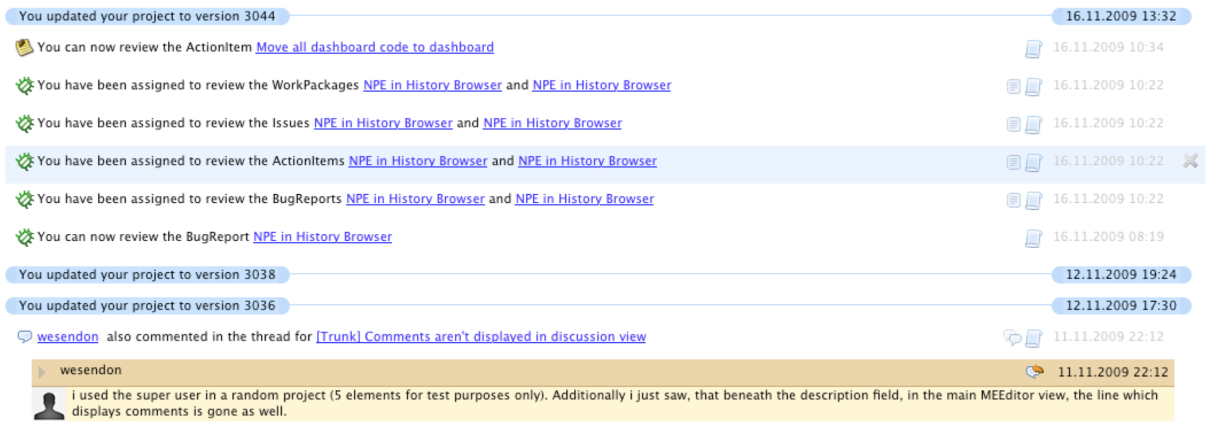

History Browser

History for Objektsteuerung

```
Set description in FunctionalRequirement "Objektsteuerung" to "(empty)" from previous value "(empty)"
Set description in FunctionalRequirement "Objektsteuerung" to "(empty)" from previous value "(empty)"
Moved FunctionalRequirement "Anzeige von Schaltplänen in 2D" from FunctionalRequirement "Objektsteuerung" to FunctionalRequirement "3D Ansicht"
Added FunctionalRequirement "Objektsteuerung" to annotatedModelElements in ActionItem "AIR41 Periodische Temperatur U..."
Added FunctionalRequirement "Objektsteuerung" to annotatedModelElements in ActionItem "AIR45 UI-Verbesserungen für No..."
Added FunctionalRequirement "Objektsteuerung" to annotatedModelElements in ActionItem "Aktionen über Netzwerk schicke..."
Moved FunctionalRequirement "Anzeige von Schaltplänen in 3D" from FunctionalRequirement "Objektsteuerung" to FunctionalRequirement "3D Ansicht"
Removed FunctionalRequirement "Anzeige von Schaltplänen in 3D" as child in FunctionalRequirement "Objektsteuerung"
Changed refinedRequirement from FunctionalRequirement "Objektsteuerung" to FunctionalRequirement "Interaktion mit Objekten" for "(unkown element)
Set cost in FunctionalRequirement "Objektsteuerung" to "100" from previous value "0"
Changed refinedRequirement from FunctionalRequirement "Objektsteuerung" to (unkown element) for "(unkown element)
Added 3 FunctionalRequirements as children in FunctionalRequirement "Objektsteuerung"
Moved FunctionalRequirement "Markierung von Objekten" from FunctionalRequirement "Objektsteuerung" to FunctionalRequirement "3D Ansicht"
Added 5 FunctionalRequirements as children in FunctionalRequirement "Objektsteuerung"
Moved FunctionalRequirement "Anzeige von Schaltplänen in 2D" from FunctionalRequirement "Interaktion mit Objekten" to FunctionalRequirement "Objektsteuerung"
Moved FunctionalRequirement "Textuelle Suche nach Objekten" from FunctionalRequirement "Objektsteuerung" to FunctionalRequirement "Objektsuche"
Added (unkown element) as child in FunctionalRequirement "Objektsteuerung"
Added (unkown element) as child in FunctionalRequirement "Objektsteuerung"
Added FunctionalRequirement "Textuelle Suche nach Objekten" as child in FunctionalRequirement "Objektsteuerung"
Added FunctionalRequirement "Markierung von Objekten" as child in FunctionalRequirement "Objektsteuerung"
Added FunctionalRequirement "Objektdetails anzeigen" as child in FunctionalRequirement "Objektsteuerung"
Added (unkown element) as child in FunctionalRequirement "Objektsteuerung"
Added (unkown element) as child in FunctionalRequirement "Objektsteuerung"
Added FunctionalRequirement "Objektsteuerung" as child in LeafSection "Functional requirements"
Created FunctionalRequirement "Objektsteuerung"
```

2.3.4 Dashboard

The Dashboard shows notifications for recent changes on the project that might be of interest for the current user. It is automatically invoked on each update but can be also opened from the project context menu. For each notification you can click on the history button to show the change the notification originated from in the history browser. Some notifications can be expanded to show more details by clicking them. Notifications can also be sent manually during commit to explicitly notify a user about a change. Additionally users can subscribe to model elements in the context menu of each model element. They will then receive a notification if the element changes. Furthermore the dashboard displays notifications if you were assigned a task or if the context of one of your tasks changed.


```
You updated your project to version 3044                                                        16.11.2009 13:32
You can now review the ActionItem Move all dashboard code to dashboard                           16.11.2009 10:34
You have been assigned to review the WorkPackages NPE in History Browser and NPE in History Browser   16.11.2009 10:22
You have been assigned to review the Issues NPE in History Browser and NPE in History Browser    16.11.2009 10:22
You have been assigned to review the ActionItems NPE in History Browser and NPE in History Browser   16.11.2009 10:22
You have been assigned to review the BugReports NPE in History Browser and NPE in History Browser    16.11.2009 10:22
You can now review the BugReport NPE in History Browser                                          16.11.2009 08:19
You updated your project to version 3038                                                        12.11.2009 19:24
You updated your project to version 3036                                                        12.11.2009 17:30
wesendon also commented in the thread for [Trunk] Comments aren't displayed in discussion view   11.11.2009 22:12
  wesendon                                                                                       11.11.2009 22:12
  i used the super user in a random project (5 elements for test purposes only). Additionally i just saw, that beneath the description field, in the main MEEditor view, the line which
  displays comments is gone as well.
```

2.3.5 Document Export

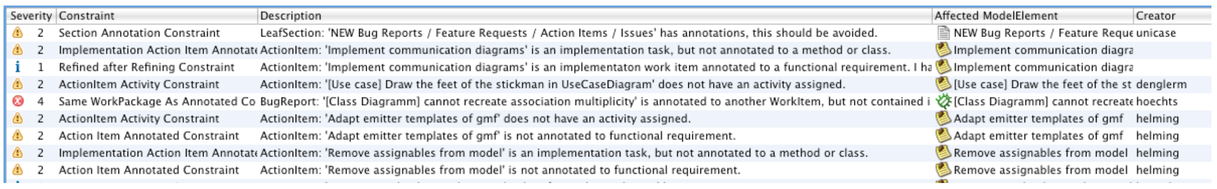
The document export feature allows exporting models to documents. You can invoke it on any model element from its context menu and it will export the element and all its contained elements. Exporting a composite section will create a document with the sub structure of the respective section and containing all elements of the respective leaf sections. Exporting a meeting is a special case and will create a meeting agenda or protocol as discussed in the meeting model section. You can choose to export to the Portable Document Format (PDF) or the Rich Text Format (RTF).

2.3.6 Code Generation

UNICASE can be used to generate code from an object model. The code generation can be invoked from the context menu of an implementation package. The package and its content are exported to an EMF Ecore file, which can be used to generate code from with the Eclipse Modeling Framework or the Open Architecture Ware Frameworks.

2.3.7 Validation View

The validation view allows validating the models in a project. It can be invoked from the context menu of project. After validation is complete it will display a number of validation violations. By double clicking a validation violation the model element where the violation occurs is opened and the respective field of the model element is flagged in red. It is good practice to assign a validation master for each team who is responsible for keeping the models of a team free of validation errors. Please note that the validation master is only responsible for making sure that all validation errors are removed but not for removing the same by him- or herself.



Severity	Constraint	Description	Affected ModelElement	Creator
⚠ 2	Section Annotation Constraint	LeafSection: 'NEW Bug Reports / Feature Requests / Action Items / Issues' has annotations, this should be avoided.	NEW Bug Reports / Feature Reque	unicase
⚠ 2	Implementation Action Item Annotat	ActionItem: 'Implement communication diagrams' is an implementation task, but not annotated to a method or class.	Implement communication diagra	
ℹ 1	Refined after Refining Constraint	ActionItem: 'Implement communication diagrams' is an implementaton work item annotated to a functional requirement. I ha	Implement communication diagra	
⚠ 2	ActionItem Activity Constraint	ActionItem: '[Use case] Draw the feet of the stickman in UseCaseDiagram' does not have an activity assigned.	[Use case] Draw the feet of the st	denglerm
⊗ 4	Same WorkPackage As Annotated Co	BugReport: '[Class Diagramm] cannot recreate association multiplicity' is annotated to another WorkItem, but not contained i	[Class Diagramm] cannot recreate	hoechts
⚠ 2	ActionItem Activity Constraint	ActionItem: 'Adapt emitter templates of gmf' does not have an activity assigned.	Adapt emitter templates of gmf	helming
⚠ 2	Action Item Annotated Constraint	ActionItem: 'Adapt emitter templates of gmf' is not annotated to functional requirement.	Adapt emitter templates of gmf	helming
⚠ 2	Implementation Action Item Annotat	ActionItem: 'Remove assignables from model' is an implementation task, but not annotated to a method or class.	Remove assignables from model	helming
⚠ 2	Action Item Annotated Constraint	ActionItem: 'Remove assignables from model' is not annotated to functional requirement.	Remove assignables from model	helming

2.4 Document Management

In the following we provide templates and guidelines for the documents that can be maintained in UNICASE:

- Requirements Analysis Document
- System Design Document
- Object Design Document
- Project Management Document

2.4.1 Requirements Analysis Document

The system design documents contains the results of the requirements analysis activity. This is a template you can use for the requirements analysis document:

- *CompositeSection*: Requirements Analysis Document
    - CompositeSection: Scenario Model
        - *LeafSection*: Scenarios
        - *LeafSection*: ActorInstances
    - *CompositeSection*: Requirements

        - *LeafSection*: FunctionalRequirements
        - *LeafSection*: Non-Functional Requirements
    - *CompositeSection*: UseCase Model
        - *LeafSection*: UseCases
        - *LeafSection*: Actors

The analysis object model is part of the subsystem decomposition in the system design document in the next section.

2.4.2 System Design Document

The system design documents contains the results of the system design activity. This is a template you can use for the System Design Document:

- *CompositeSection*: System Design Document
    - *LeafSection*: Newly identified classes
    - *LeafSection*: Subsystem Decomposition

The System Design Document is sparse in UNICASE. All other parts of a system design document are more of a sequential and text-based nature and cannot be effectively managed in UNICASE. You can export the subsystem decomposition to RTF with the document export feature and then compose the remainder of the document in a word processor.

When classes are identified from the use case model or during analysis in general they can be added to the newly identified classes section until a destination package and subsystem have been selected for them.

The subsystem decomposition consists of a number of packages representing the subsystems and the classes and sub-packages that reside in the subsystems. A subsystem should always declare a façade class to represent its interface.

2.4.3 Object Design Document

The Object Design document contains the results from the Object Design activity. This is a template you can use for the object design document:

- *CompositeSection*: Object Design Document
    - *LeafSection*: Object Design Model

The Object Design Document is sparse in UNICASE. All other parts of an object design document are more of a sequential and text-based nature and cannot be effectively managed in UNICASE. You can export the object model to RTF with the document export feature and then compose the remainder of the document in a word processor.

The object model consists of a number of implementation packages and classes. These can be transformed into code by the code generation feature of UNICASE as discussed earlier. In general it is only useful to maintain an explicit object design model if code is generated from this model; otherwise the object model should be part of the source code documentation.

2.4.4 Project Management Document

In the Project Management Document important information about the project and its processes and structure is contained. This is a template you can use for the project management document:

- *CompositeSection*: Project Management Document
    - CompositeSection: Organizational Model
        - *LeafSection*: Users
        - *LeafSection*: Groups
    - *CompositeSection*: Iteration Planning
        - *LeafSection*: New work items
        - *LeafSection*: Open work (Backlog)
        - *LeafSection*: Iterations (Sprints)
    - *LeafSection*: Roadmap
    - *LeafSection*: Meetings

The organizational model can be modeled as groups and their members. In Iteration Planning new work items that have not accepted as to be done yet can be put in the "New work items" section. The open work section holds work, that is to be done, but that has not been planned yet. Normally a manager will inspect the incoming work in new work items and then add them to the open work section from time to time. Finally the iterations section holds a number of work packages modeling iterations that contain the work items that are planned for it. In the Scrum methodology the open work section is often referred to as Backlog while the iterations are called Sprints.

In Roadmap milestones can be used to define a plan of which requirements or use cases are to be completed in the future and at what milestone.

Finally meetings can be put in the meetings section to record meetings and their contents.