Kobold
Product Line Management System

Version 2.0

Kobold
Produktlinien-Manager

User Manual

Version history

- Version 1.0 (2004-05-18)
- Version 2.0 (2004-06-29)

Contents

1 Introduction

1.1 About this document

This manual provides instructions for the installation and usage of the product line management system Kobold.

1.2 The Kobold System

The primary purpose of Kobold is to provide a tool for the development and maintenance of software product lines. Kobold also supports the establishment of a role-based development process.

1.3 Structure of this document

This user manual will help you to install the Kobold system and to determine the technical conditions.

The description of the usage is divided in two parts. The first part will provide descriptions of each window and each menu item.

The second part will describe the functionality of Kobold. This tutorial will help you answer your questions while working with Kobold.

2 Technical Requirements

Kobold runs on

- Windows NT/2000/XP
- Linux (Motif)
- Linux (GTK 2)
- Solaris 8 (SPARC/Motif)
- QNX (x86/Photon)
- AIX (PPC/Motif)
- HP-UX (HP9000/Motif)
- Mac OSX (Mac/Cocoa)

with the following requirements:

- 400 MHz
- 256 MB main memory
- 200 MB fixed-disk storage
- Java 2 version 1.4
- Eclipse 3.0

Its server is purely based on Java and therefore runs on:

- Solaris-SPARC
- Windows NT/2000/XP
- Linux

3 Instructions for Installation

3.1 Checking out the files

After you have started Eclipse 3.0, change to CVS Repositories perspective and add a CVS repository.

Enter the following data (see 3.1):

- host: cvs.berlios.de
- repository path: /cvsroot/kobold
- user: anonymous

Leave the remaining data untouched and confirm the dialog.

Open the tree along HEAD, kobold and src (see 3.2). Check out the five folders kobold.client.plam, kobold.client.vcm, kobold.client.serveradmin, kobold.common and kobold.server through 'check out as..' (context menu) and confirm with 'Finish'.

Open the 'Window' menu and select 'preferences'. Select 'plug-in development' and 'Target Platform' and press the button 'Not in Workspace'. Confirm with OK.

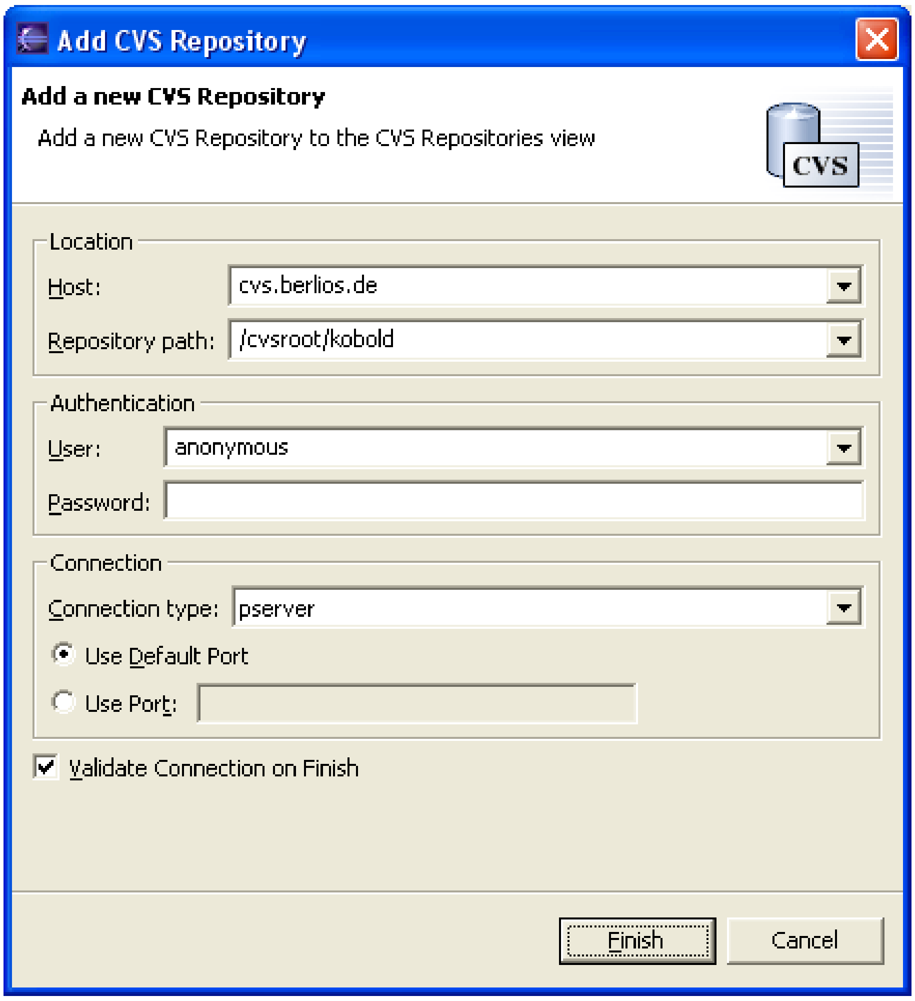
Projects kobold.client.plam, kobold.client.vcm and kobold.common:

Right-click on the project and select 'properties'. Select 'Java build path' and the 'source' tab. Remove the existing entry and add a new one by pressing 'Add Folder'. Choose the src-folder and confirm. Append '/bin' to the default output folder (see 3.3). Close the properties dialog.

Project kobold.server:

Right-click on the project and select 'properties'. Select 'Java build path' and the 'source' tab. Remove the existing entry and add a new one by pressing 'Add Folder'. Choose the src-folder and confirm. Append '/bin' to the default output folder. Switch to the 'projects' tab and select 'kobold.common'. Finally, switch to the 'libraries' tab and press the 'add jars...' button. A dialog opens which displays the project kobold.common. Open the tree and select every jar-file in the contrib-folder and in the folders contained in contrib. Confirm with 'OK' and close the properties dialog.

Again, right-click on the project kobold.client.plam and select 'PDE Tools' and then 'update classpath'. Select kobold.client.plam, kobold.client.vcm and kobold.common and



Add a new CVS Repository

Add a new CVS Repository to the CVS Repositories view

Location

Host: cvs.berlios.de

Repository path: /cvsroot/kobold

Authentication

User: anonymous

Password:

Connection

Connection type: pserver

[✓ Use Default Port]

Use Port:

[✓ Validate Connection on Finish]

Figure 3.1: Adding a new repository

confirm.

3.2 Setting the correct properties

First of all edit the server.properties file from *kobold.server/server.properties* to suite your local requirements. Under UNIX this file might look similiar to the following bits:

```
# Note storePath is the prefix path for all kobold stores!
kobold.server.storePath=/tmp/
kobold.server.globalMessageStore=global.xml
kobold.server.messageStore=messages.xml
kobold.server.productStore=product.xml
```

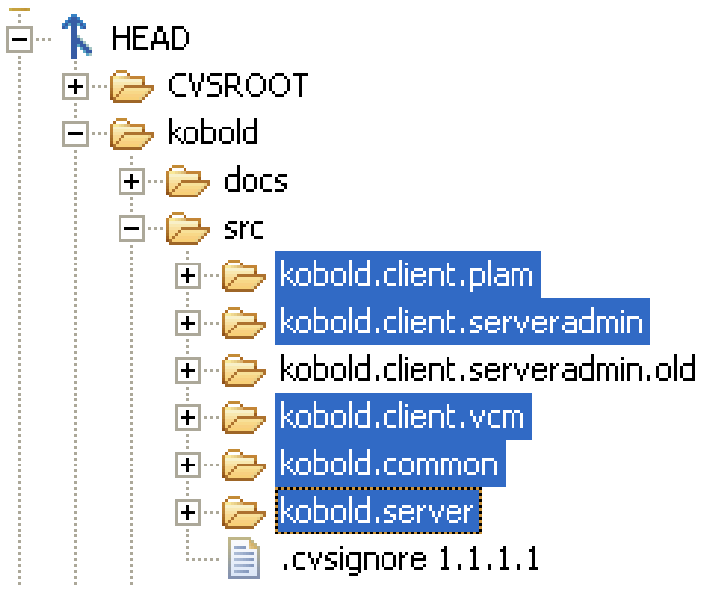


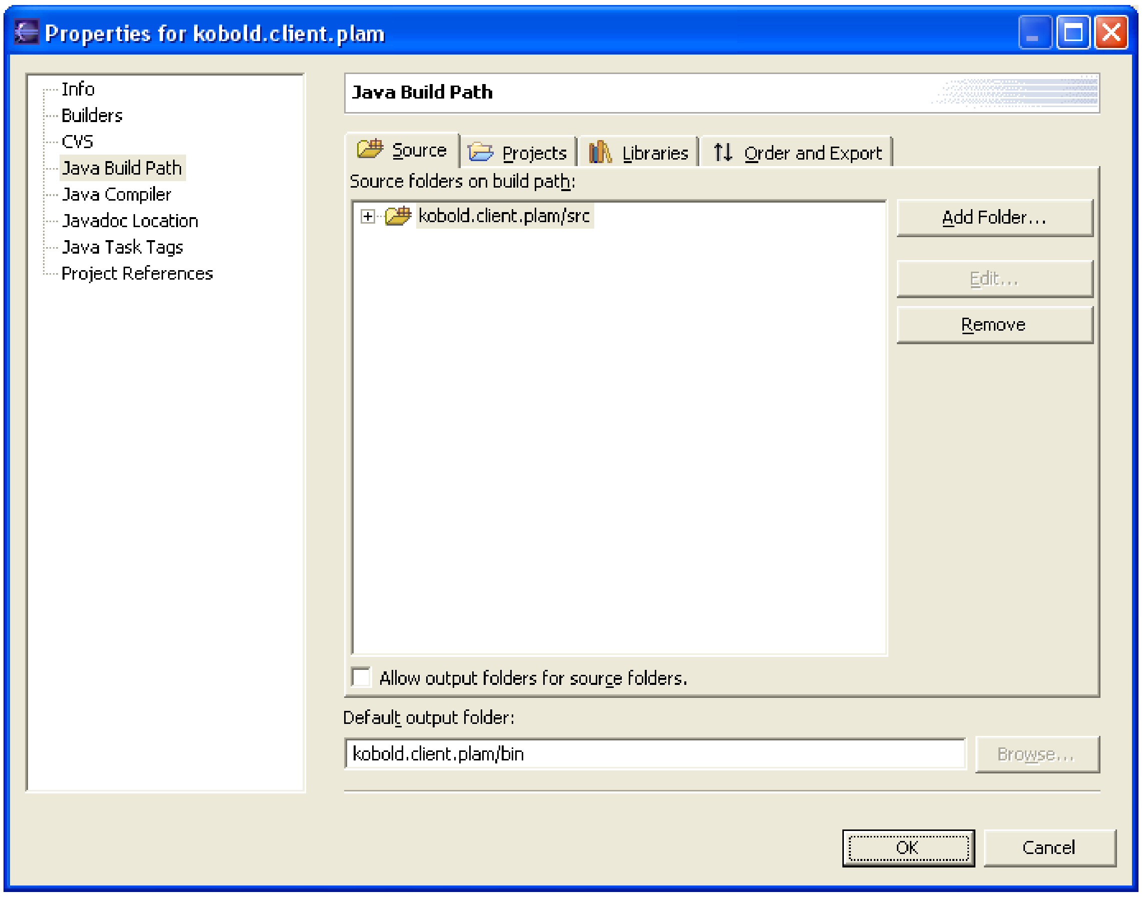
Figure 3.2: This is how your tree should look like

```
kobold.server.userStore=user.xml
#
java.protocol.handler.pkgs=com.sun.net.ssl.internal.www.protocol
javax.net.debug=all
javax.net.ssl.keyStore=../kobold.common/scripts/keystore
javax.net.ssl.keyStorePassword=kobold1
javax.net.ssl.trustStore=../kobold.common/scripts/truststore
javax.net.ssl.trustStorePassword=kobold1
```

Under Windows-based systems you've to change the paths into a DOS-alike format.

In the following table all properties are described in detail:

Property	**Description**
kobold.server.storePath	destination path for files stored by the server
kobold.server.globalMessageStore	file name to store global messages
kobold.server.messageStore	file name to store pending messages
kobold.server.productStore	file name to store products and productlines
kobold.server.userStore	file name to store user data
java.protocol.handler.pkgs	default protocol to commincate
javax.net.debug	debug level of net communication
javax.net.ssl.keyStore	path to your SSL keystore
javax.net.ssl.keyStorePassword	password to access your SSL keystore
javax.net.ssl.trustStore	path to your SSL truststore
javax.net.ssl.trustStorePassword	password to access your SSL truststore



Source folders on build path:
kobold.client.plam/src

Allow output folders for source folders.

Default output folder:
kobold.client.plam/bin

Figure 3.3: This is how your buildpath should look like

You also have to edit the sat.properties file from *kobold.client.serveradmin/sat.properties* to suite your local requirements. Under UNIX this file might look similiar to the following bits:

```
#used to communicate with Kobold servers via ssl
java.protocol.handler.pkgs=com.sun.net.ssl.internal.www.protocol
javax.net.debug=all
javax.net.ssl.keyStore=/home/garbeam/eclipse/kobold.common/scripts/keysto
javax.net.ssl.keyStorePassword=kobold1
javax.net.ssl.trustStore=/home/garbeam/eclipse/kobold.common/scripts/trus
javax.net.ssl.trustStorePassword=kobold1
```

As you can see, it contains the same properties as the *server.properties* file for SSL communication, but since the Kobold Serveradmin Tool is independent from the server it has its own properties.

Right after your first start of the Kobold Client Feater Set, change the Kobold preferences so that the client-server communication works properly. For more information go to the tutorial section.

4 The concepts in Kobold

4.1 Productline

Kobold is a tool to administrate productlines. By specifying an architecture in which core assets are linked with each other through dependency and exclusion edges, the product line engineer creates a basis for all the products of the product line.

4.2 Component

Components are abstract modules in the productline architecture. Instances of them (i.d. variants and custom components) are used for the different products of the productline.

4.3 Custom Component

Custom components are instances of components and used in product architectures. They can differ from the corresponding components in order to fit the product.

4.4 Variant

Variants are modifications of components.

4.5 Release

Releases are created to save the current state of a variant.

4.6 Meta Node

Meta nodes are a means to creating complex relationships between nodes. They represent the relationships AND and OR.

4.7 Dependency Edge

A dependency edge indicates that the nodes connected by the edge depend on each other. Therefore one of them can't be used without the other.

4.8 Exclusion Edge

An exclusion edge indicates that the nodes connected by the edge can't be used in the same product.

4.9 GXL

GXL is a graph exchange language and used to export the architecture, etc. For more information: http://www.gupro.de/GXL/

4.10 Product Line Engineer (PLE)

The product line engineer has the responsibility for the product line.

4.11 Product Engineer (PE)

The product engineer administrates a specific product and is a subordinate of the product line engineer.

4.12 Product Developer (P)

Product developers are subordinates of the product engineer. They develop the software product.

4.13 Workflow

A workflow is a working process. You can create them in order to trigger certain activities when the server executes one of the following commands:

- login
- logout
- get roles
- get product role
- add user
- get productline
- get product
- add product
- add role
- remove role
- apply productline modifications
- apply product modifications
- remove user

5 User Interface

5.1 Main Window

In the Main Window you can see the following editors and views (see 5.1):

- Architecture Editor
- Role View
- Workflow View
- Minimap

The menu offers you in addition to the Eclipse standard options the following possibilities:

- Checking out a product
- Changing Kobold properties

Look up the explanations in the ”Tutorial” section.

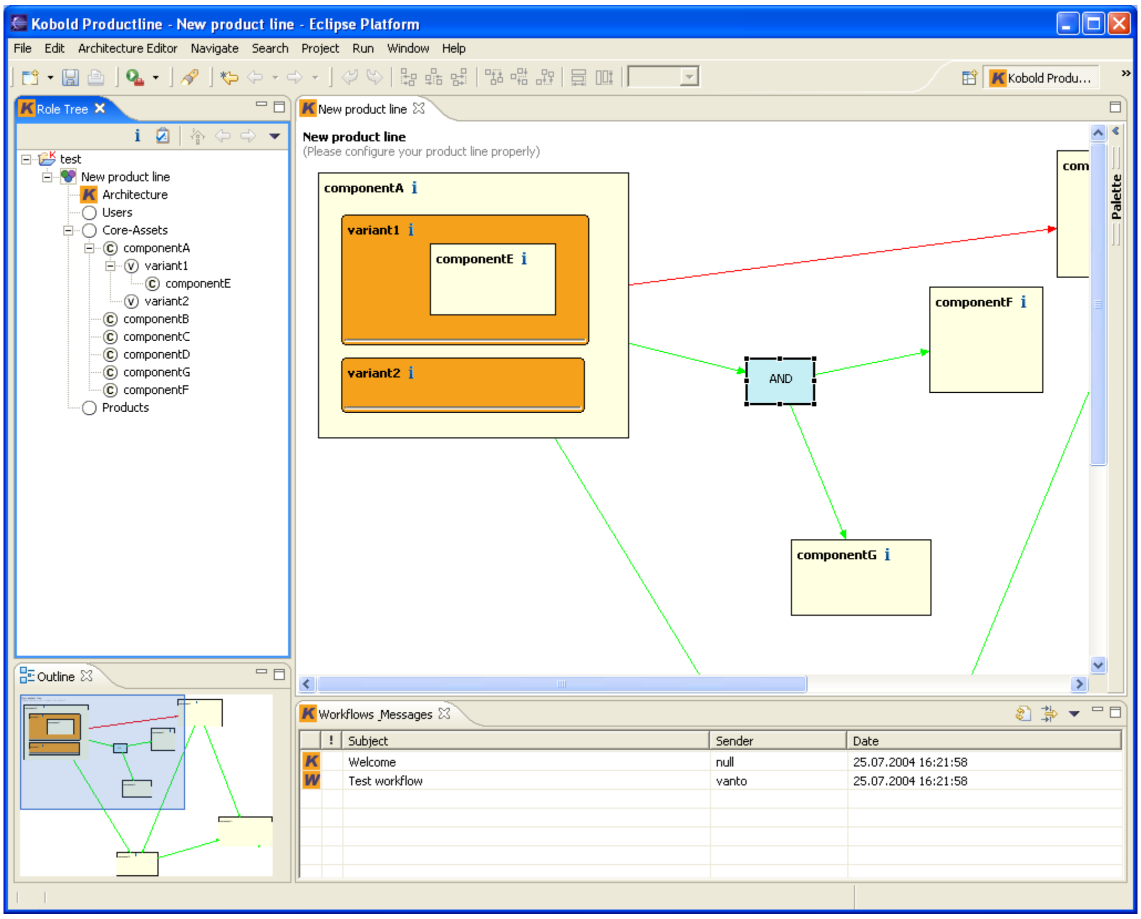
5.2 Architecture Editor

The Architecture Editor (see 5.2) displays the architecture of the product, or product line respectively, that is selected in the Role View.

Architectures are directed graphs that consist of nodes and edges. Nodes represent components, variants and releases. Directed edges represent any relationship between nodes.

The Architecture Editor offers you the following options:

- Creating a variant
- Creating a component
- Creating a release
- Creating a meta node
- Deleting a component, variant or release



!	Subject	Sender	Date
K	Welcome	null	25.07.2004 16:21:58
W	Test workflow	vanto	25.07.2004 16:21:58

Figure 5.1: Main Window

- Deleting a meta node
- Creating a dependency edge
- Creating an exclusion edge
- Deleting an edge
- Marking a component, variant or release ”deprecated”
- Setting the maintainers of a component
- Selecting resources for variants and releases
- Moving an item
- Changing the size of an item
- Composing a product
- Export

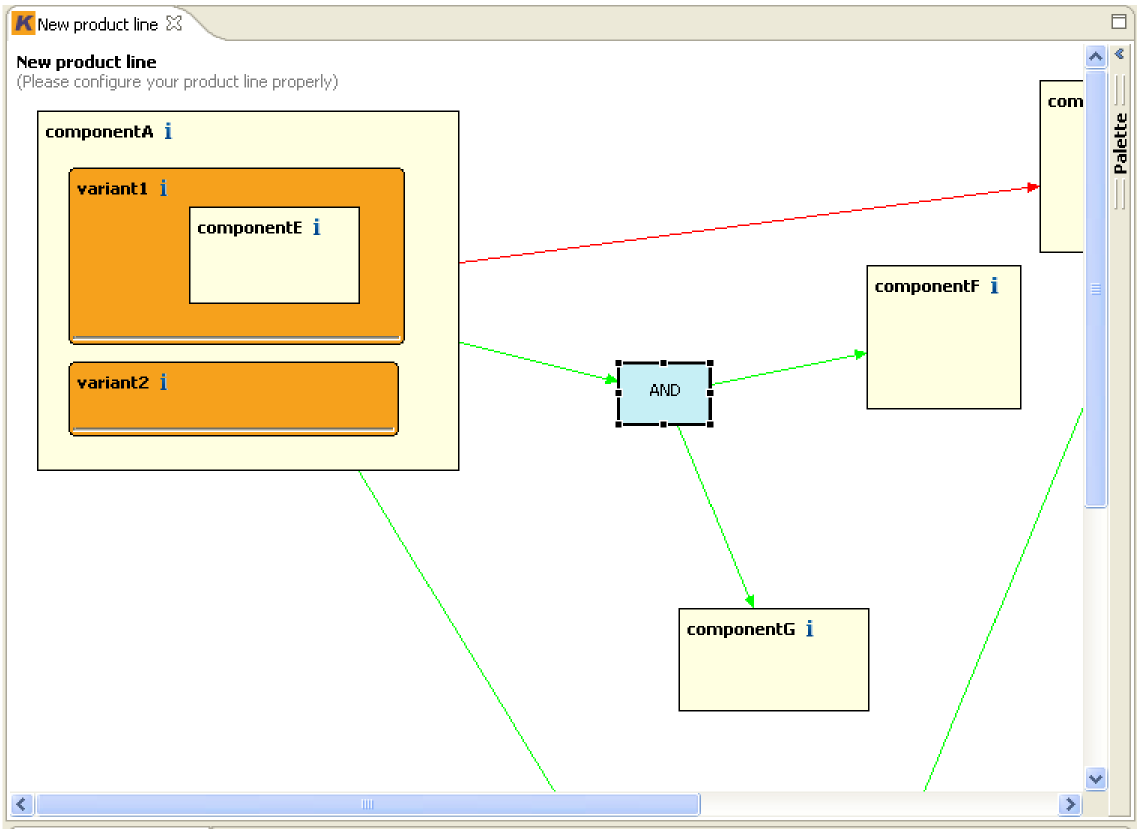


Figure 5.2: Architecture Editor

Most of them can be reached through the palette (see 5.3), that opens up when you click on it.

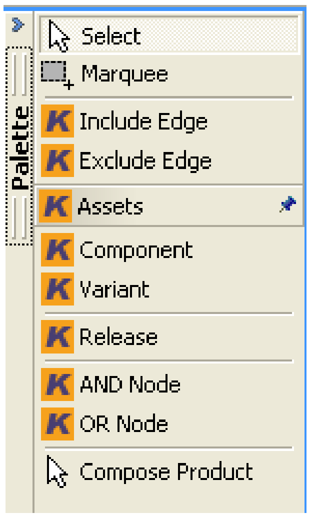


Figure 5.3: The palette

Please look up the explanations in the ”Tutorial” section.

5.3 Role View

The Role View shows the current projects that are checked out in your workspace (see 5.4). A project is usually a product line. You can navigate between the different projects by selecting them. If you open the tree, you see the name of the product line. Double-click on ’Architecture’ and the Architecture Editor will open. The open tree also displays all products, PLEs and core assets that belong to the productline.

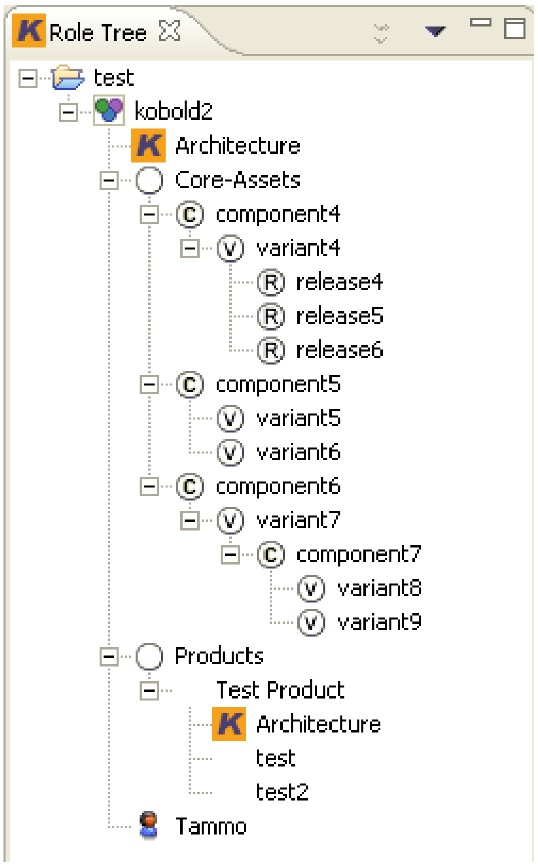


Figure 5.4: Role View

When you right-click on a project, a context menu will open where you can choose between different actions (see 5.5). Among those are:

- Creating a user
- Deleting a user
- Change one’s password

- Update full name
- Configure asset
- Suggesting an asset for a Core Group
- Generate a metainfo document
- Export

Look up the explanations in the ”Tutorial” section.

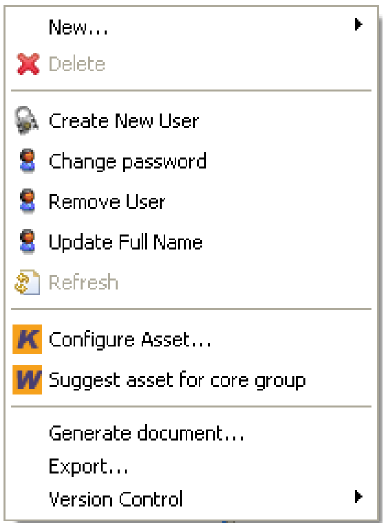
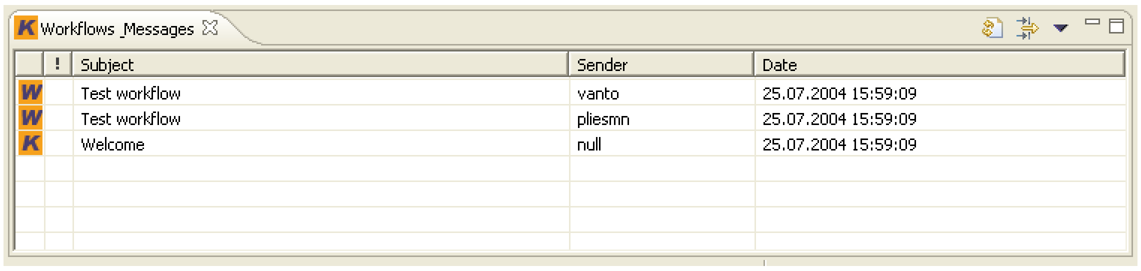


Figure 5.5: Role Tree Context Menu

5.4 Workflow View

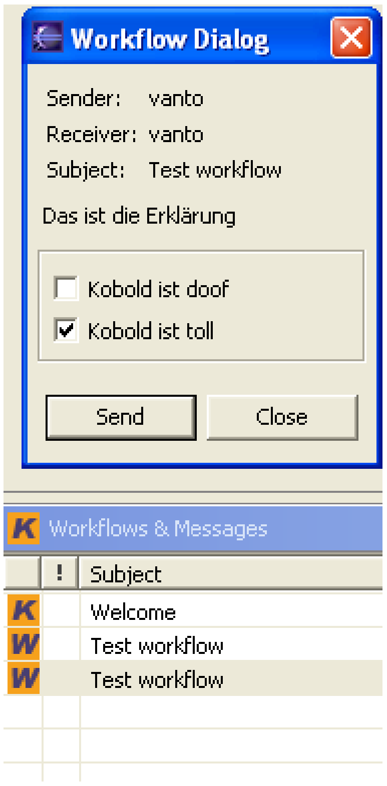
In the bottom of the window you can see the Workflow View which displays all the messages and workflows for the current user (see 5.6).



	!	Subject	Sender	Date
W		Test workflow	vanto	25.07.2004 15:59:09
W		Test workflow	pliesmn	25.07.2004 15:59:09
K		Welcome	null	25.07.2004 15:59:09

Figure 5.6: Workflow/Task View

Double-click on an entry and a separate dialog will be opened where you can read the details of that message or workflow (see 5.7).



Sender: vanto
Receiver: vanto
Subject: Test workflow

Das ist die Erklärung

☐ Kobold ist doof
☑ Kobold ist toll

!	Subject
	Welcome
	Test workflow
	Test workflow

Figure 5.7: Message Dialog

When you right-click on a message, a context menu will open where you can choose between different actions (see 5.8). Among these are:

- Fetching messages
- Deleting the selected message

Look up the explanations in the ”Tutorial” section.

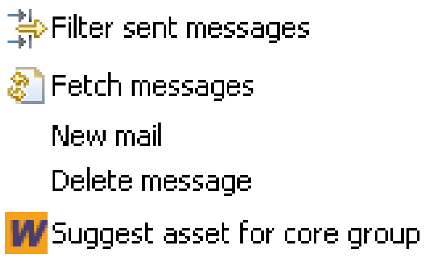


Figure 5.8: Message Context Menu

The Workflow View offers you the following additional options:

- Writing a mail

- Answering a mail
- Suggesting an asset for a Core Group
- Dealing with a Core Group suggestion

Look up the explanations in the ”Tutorial” section.

5.5 Minimap

The Minimap (see 5.9) is on the left side beneath the Role View. It shows the whole architecture. By clicking at one spot on the map, the Architecture Editor automatically centers on that spot.

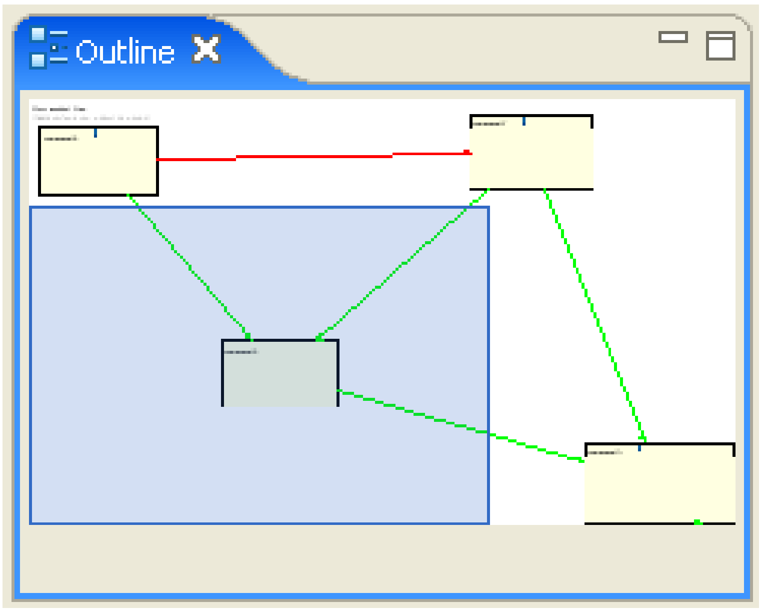


Figure 5.9: Minimap

5.6 Server Administration Tool (SAT)

The Server Administration Tool provides you with an interface to your server.

During the starting process you have to enter the URL and password of the server you want to administrate. Once the tool is started, you have the following possibilities:

- create a product line
- delete a product line

- upgrade an existing user to PLE status
- remove PLE rights
- get a list of all existing commands
- create a new user
- delete a user
- get a list of all productlines
- get a list of all PLEs of a productline
- get a list of all users
- exit the tool

6 Tutorial

6.1 Getting started

6.1.1 Kobold Server

To start the Kobold server within Eclipse just select ’Run...’ in the ’Run’ menu from Eclipse and select the class *kobold.server.SecureKoboldWebServer* with a numerical programm argument for the port it should listen for connections. You’ve to enter only one VM argument which is needed to locate the *server.properties* file and to suite your local requirements:

```
-Dkobold.server.configFile=/home/garbeam/eclipse/
kobold.server/server.properties
```

Make sure that your *$CLASSPATH* environment contains all JARs provided by *kobold.common/contrib* beside a Java2 JRE and the Sun JSSE classes (included by Sun JDK 1.4).

6.1.2 Kobold Client Feature Set

To start the Kobold client feature set from within Eclipse (which is currently the only way) just select ’Run...’ in the ’Run’ menu and double-click ’Run-time Workbench’. A new configuration is created which you can name as you wish (see 6.1).

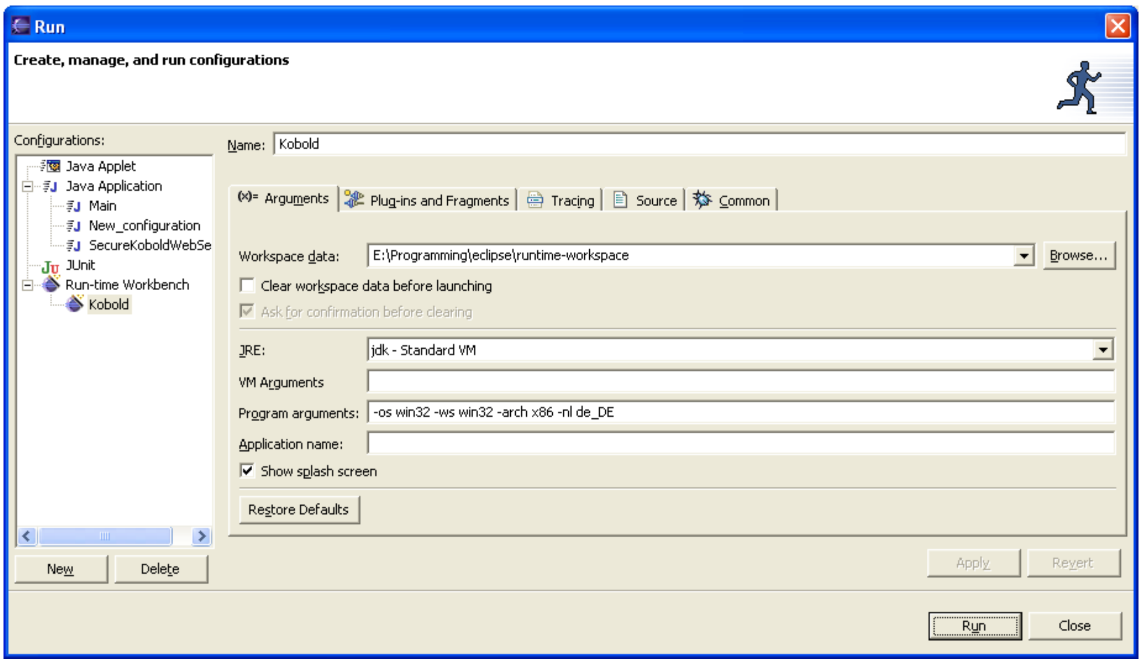
Confirm with ’Run’.

A new Eclipse instance will open with the Kobold feature set enabled.

6.1.3 Kobold Server Administration Tool

To start the Kobold Server Administration Tool within Eclipse just select ’Run...’ in the ’Run’ menu from Eclipse and select the class *kobold.client.serveradmin.ServerAdministrationTool*.

You’ve to enter only one VM argument which is needed to locate the *sat.properties* file and to suite your local requirements:


Run

Create, manage, and run configurations

Configurations:
- Java Applet
- Java Application
  - Main
  - New_configuration
  - SecureKoboldWebSe
- JUnit
- Run-time Workbench
  - Kobold

Name: Kobold

Workspace data: E:\Programming\eclipse\runtime-workspace

☐ Clear workspace data before launching

☑ Ask for confirmation before clearing

JRE: jdk - Standard VM

VM Arguments:

Program arguments: -os win32 -ws win32 -arch x86 -nl de_DE

Application name:

☑ Show splash screen

Figure 6.1: The Run dialog

```
-Dkobold.sat.configFile=/home/garbeam/eclipse/
kobold.client.serveradmin/sat.properties
```

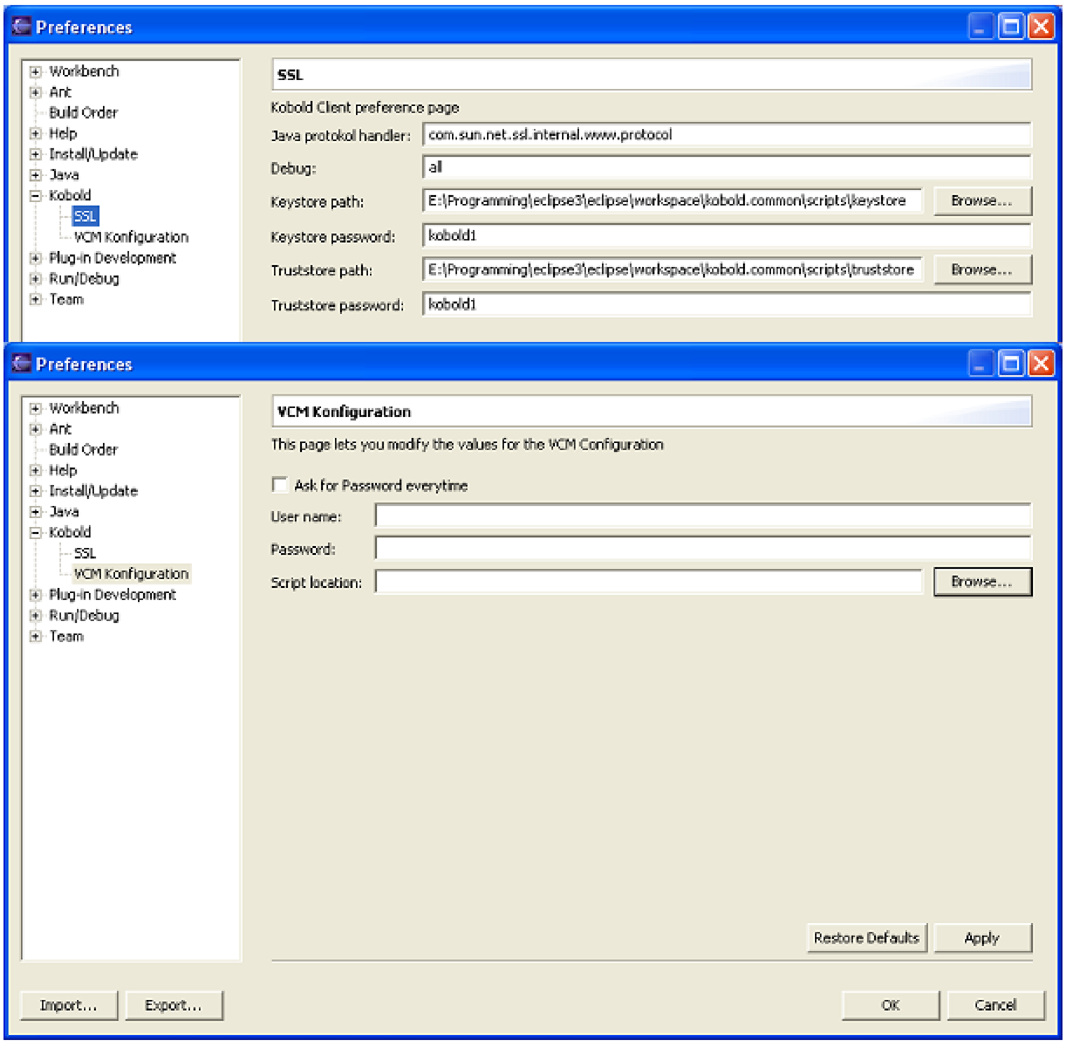
6.2 Setting the Kobold preferences

In the Kobold Client menu, select ’Window’ and then ’Preferences’. The preferences dialog opens (see 6.2). Open the Kobold tree. In the SSL tab you can enter the properties for the Kobold Client. They resemble your SAT properties (see chapter ’Installation’). The VCM Configuration allows you to save your username, password and script for the communication with the projects’ repositories.

6.3 Product-related

6.3.1 Checking out a productline

In the File menu select ’New’ and then ’Kobold PLAM Project’. The Kobold wizard opens. Enter the url of your Kobold server, your username and password. Then press ”test connection”. If the test succeeds, the ”next” button is enabled (see 6.3).



SSL

Kobold Client preference page

Java protokol handler: com.sun.net.ssl.internal.www.protocol

Debug: all

Keystore path: E:\Programming\eclipse3\eclipse\workspace\kobold.common\scripts\keystore

Keystore password: kobold1

Truststore path: E:\Programming\eclipse3\eclipse\workspace\kobold.common\scripts\truststore

Truststore password: kobold1

VCM Konfiguration

This page lets you modify the values for the VCM Configuration

☐ Ask for Password everytime

User name:

Password:

Script location:

Figure 6.2: Kobold preferences

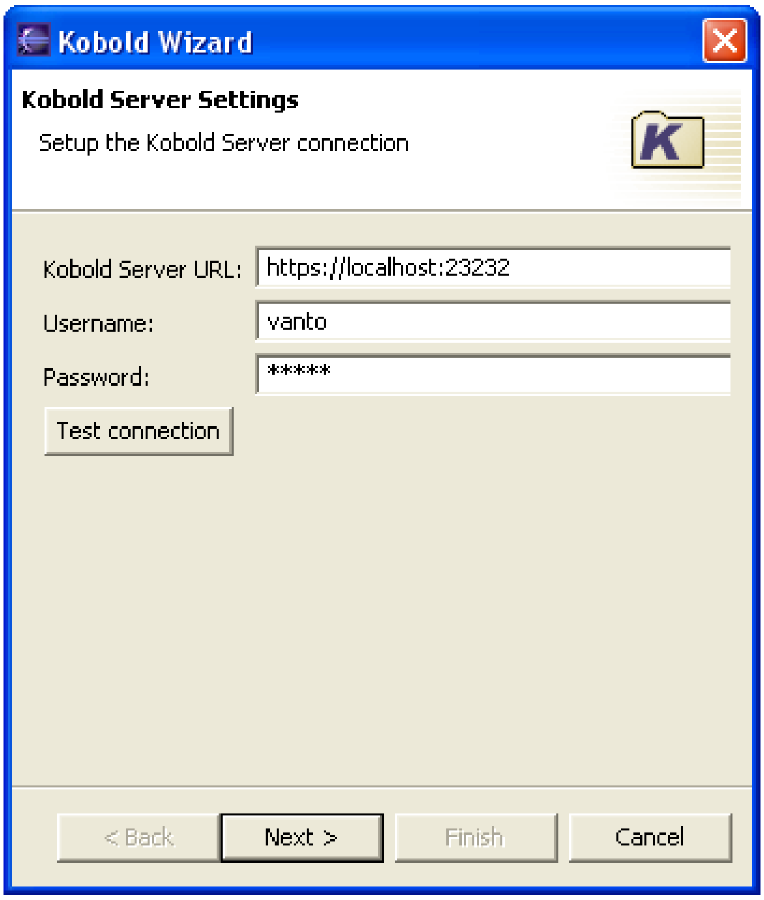
After that you choose the productline you want to check out (see 6.4).

In the last step you have to enter the name of the project you want to create (see 6.5).

A new project has been created. You can open the different views through the Window menu and 'show view'.

6.3.2 Creating a product

In the palette press 'compose product'. The components, etc. in the architecture view turn grey. You can now select the components, variants and releases you want to include into you new product. The chosen objects turn blue (see 6.6). When done press the 'create product' button in the Architecture Editor. A dialog opens where you can enter



Kobold Server URL: https://localhost:23232
Username: vanto
Password: *****

Figure 6.3: Kobold wizard

the name and metainfo of your new product. After confirming the dialog, you can see your new product in the Role View.

6.4 Node-related

6.4.1 Creating a component

In the pallete select the "component" tool. Click within a variant or top-level in the Architecture Editor. A component is inserted (see 6.7) and a dialog opens where you can enter the meta data of the component. Per drag+drop you can simply change the size of the component.

Note: Components can only be inserted top-level or into variants.

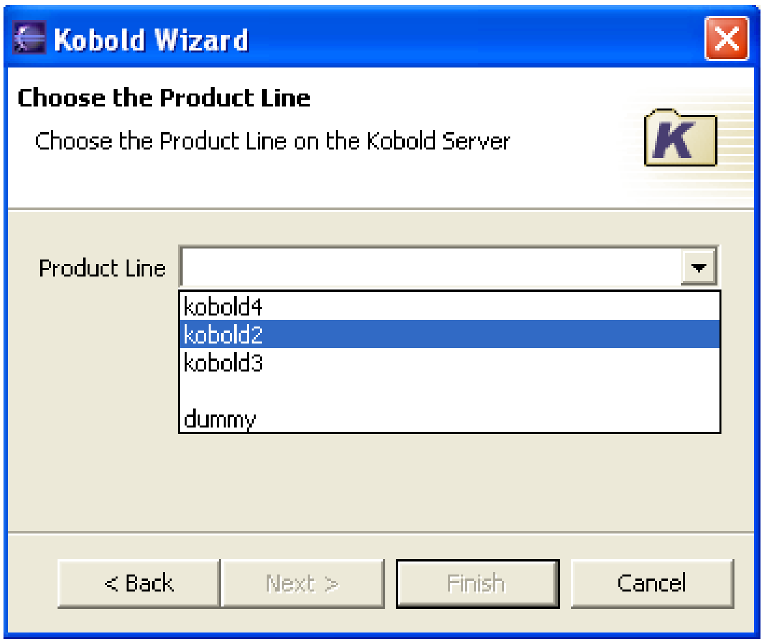


Figure 6.4: Kobold wizard

6.4.2 Creating a variant

In the pallete select the "variant" tool. Click within a component in the Architecture Editor. A variant is inserted (see 6.8) and a dialog opens where you can enter the meta data of the variant. Per drag+drop you can simply change the size of the variant.

Note: Variants can only be inserted into components.

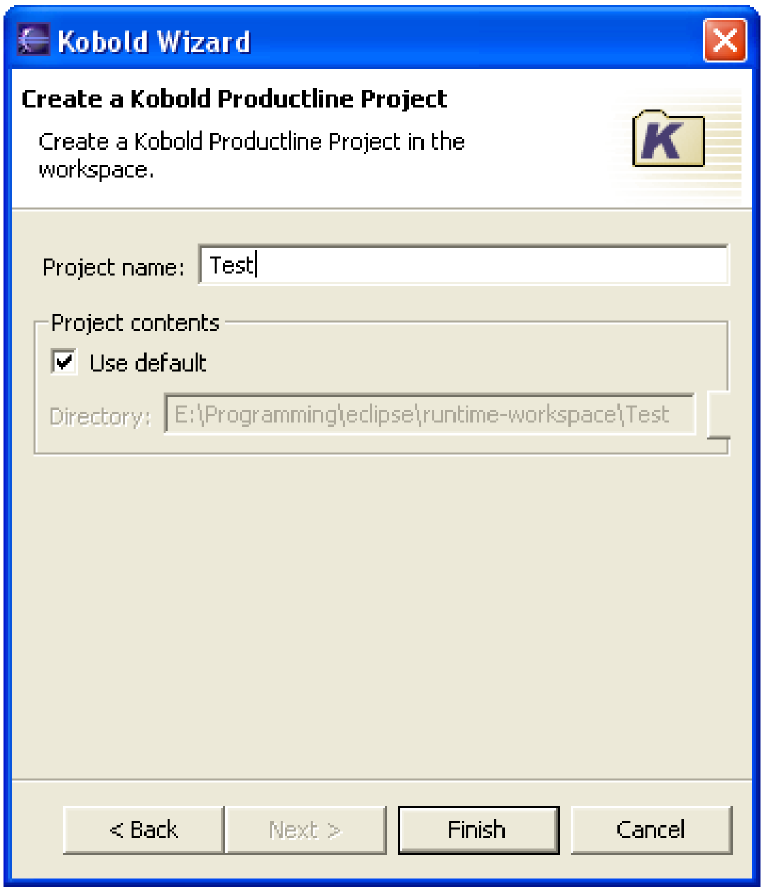
6.4.3 Creating a release

In the pallete select the "release" tool. Click within a variant in the Architecture Editor. A release is inserted (see 6.9) and a dialog opens where you can enter the meta data of the release. For each file of the variant you can select the revision number you want to add to the release. Per drag+drop you can simply change the size of the release.

Note: Releases can only be inserted into variants.

6.4.4 Creating a meta node

In the pallete select the "meta node" tool. Click within the Architecture Editor. A meta node is inserted (see 6.10). Possible meta node types are AND and OR. Be careful with the direction of the edges that are linked to a meta node! In the following architecture



Kobold Wizard

Create a Kobold Productline Project

Create a Kobold Productline Project in the workspace.

Project name: Test

Project contents

[✓] Use default

Directory: E:\Programming\eclipse\runtime-workspace\Test

Figure 6.5: Kobold wizard

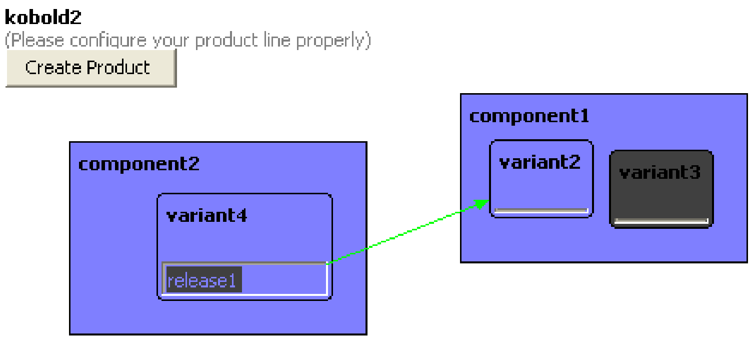


Figure 6.6: Composing a Product

component A needs components B and C. But components B and C are independent of component A (see 6.11).

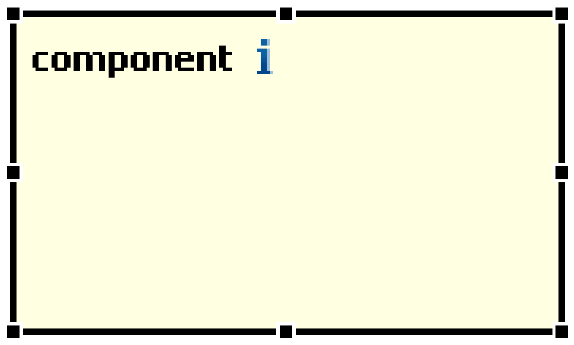


Figure 6.7: Component

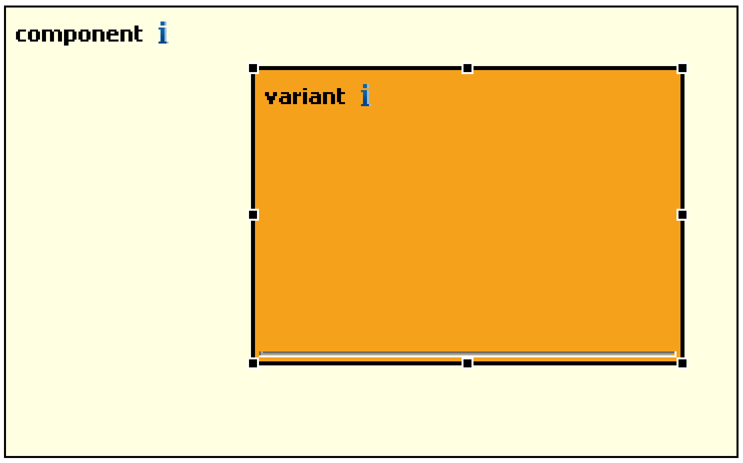


Figure 6.8: Variant

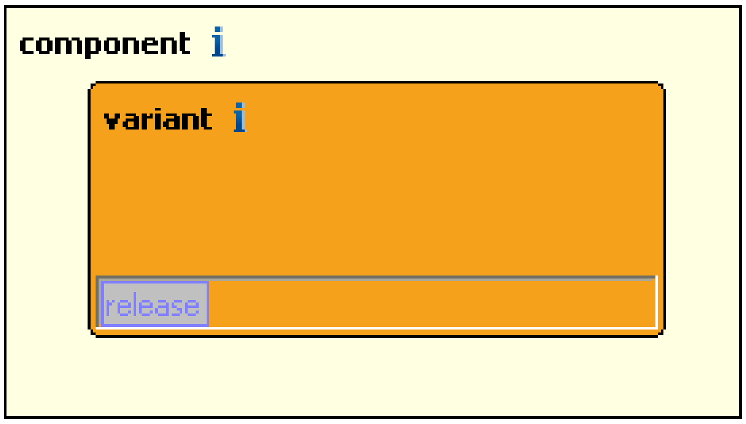


Figure 6.9: Release

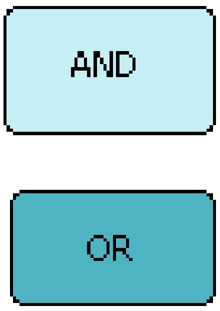


Figure 6.10: Some meta nodes

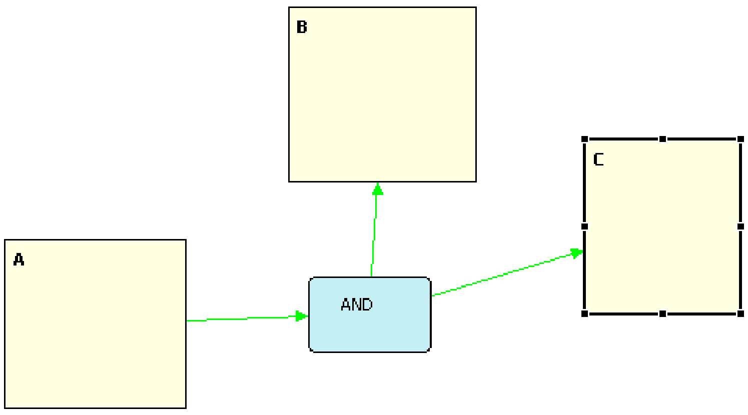


Figure 6.11: Meta node example

6.4.5 Deleting a component, variant or release

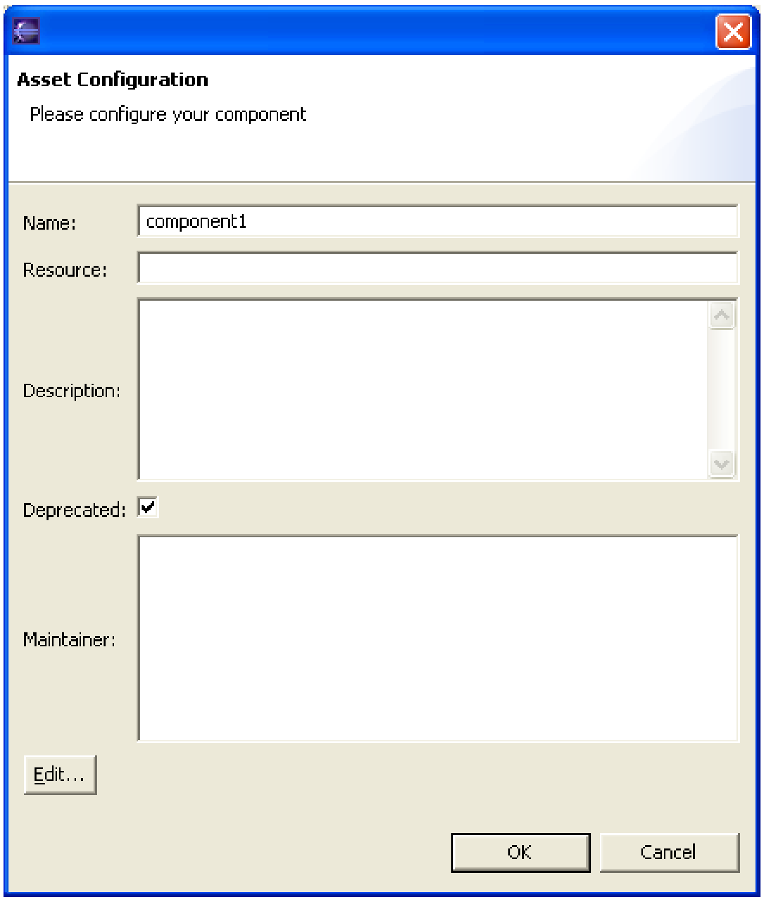
Right-click on the item you want to delete and choose ”delete” in the context menu. Alternatively you can select the item and press ”Del” on your keyboard. Is the item used in other projects you will be notified and be able to cancel the deletion process.

6.4.6 Deleting a meta node

Right-click on the meta node you want to delete and choose ”delete” in the context menu. Alternatively you can select the node and press ”Del” on your keyboard.

6.4.7 Marking a component, variant or release as ”deprecated”

Right-click on the object and choose ”configure”. The Asset Configuration dialog (see 6.12 and 6.15) opens where you can mark the check-box ’deprecated’. This sets the object on deprecated. Deprecated objects are grey with a red cross in the upper right-hand corner (see 6.13).


Asset Configuration

Please configure your component

Name: component1

Resource:

Description:

Deprecated: ✓

Maintainer:

Figure 6.12: Asset Configuration dialog (component)

6.4.8 Setting the maintainers of a component

Right-click on the component and choose ”configure”. The Asset Configuration dialog (see 6.12) opens. Press the ’Edit...’ button. A new window opens where you can select the users you want to make maintainers of the component (see 6.14). After pressing ’OK’ you can see the new maintainers in the Asset Configuration dialog.

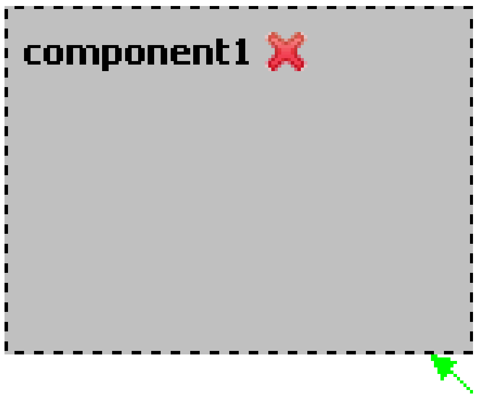
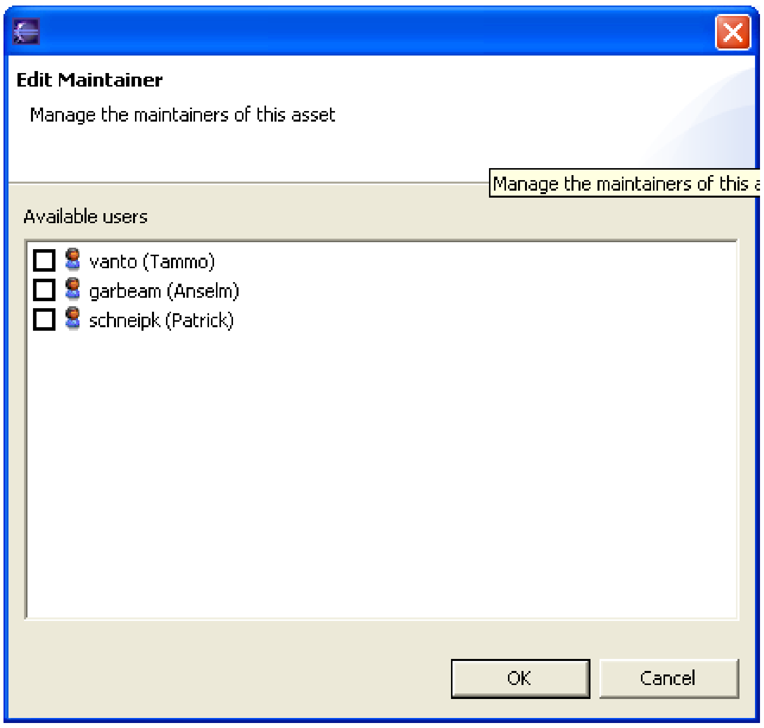


Figure 6.13: Deprecated component



Edit Maintainer

Manage the maintainers of this asset

Manage the maintainers of this a

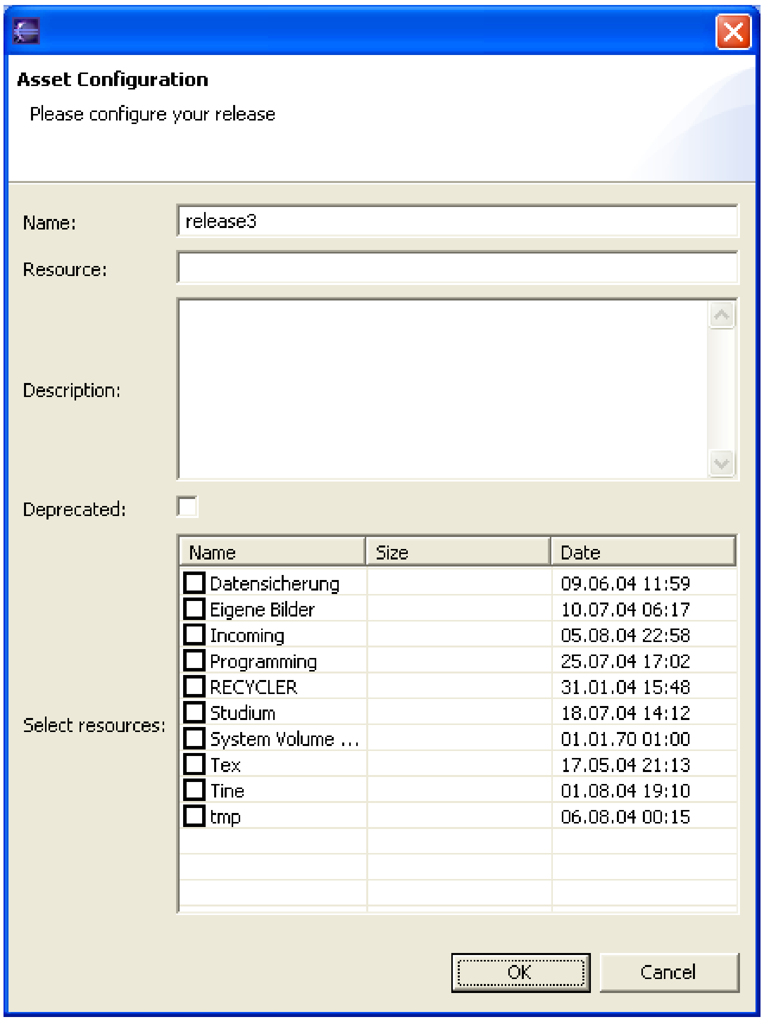
Available users

- ☐ vanto (Tammo)
- ☐ garbeam (Anselm)
- ☐ schneipk (Patrick)

Figure 6.14: Selecting maintainers

6.4.9 Selecting resources for a release

Right-click on the component and choose ”configure”. The Asset Configuration dialog (see 6.15) opens. In the bottom of the dialog you can select the files of the variant you want to include into the release.



Name: release3
Resource:
Description:
Deprecated:
Select resources:

Name	Size	Date
☐ Datensicherung		09.06.04 11:59
☐ Eigene Bilder		10.07.04 06:17
☐ Incoming		05.08.04 22:58
☐ Programming		25.07.04 17:02
☐ RECYCLER		31.01.04 15:48
☐ Studium		18.07.04 14:12
☐ System Volume ...		01.01.70 01:00
☐ Tex		17.05.04 21:13
☐ Tine		01.08.04 19:10
☐ tmp		06.08.04 00:15

Figure 6.15: Asset Configuration dialog (release)

6.4.10 Moving an item

You can move any item per drag+drop.

6.4.11 Changing the size of an item

You can change the size of an item per drag+drop.

6.5 Edge-related

6.5.1 Creating a dependency edge

In the pallete select the ”include edge” tool. Select an item in the Architecture Editor as the starting point. The next item you select will be the aiming point (see 6.16).

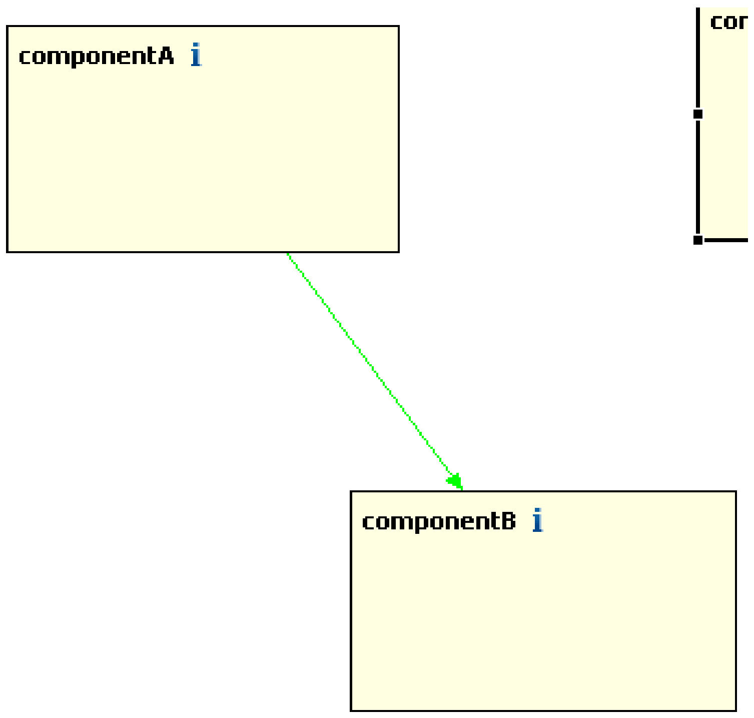


Figure 6.16: Dependency edge

6.5.2 Creating an exclusion edge

In the pallete select the ”exclude edge” tool. Select an item in the Architecture Editor as the starting point. The next item you select will be the aiming point (see 6.17).

6.5.3 Deleting an edge

Right-click on the edge you want to delete and choose ”delete” in the context menu. Alternatively you can select the edge and press ”Del” on your keyboard.

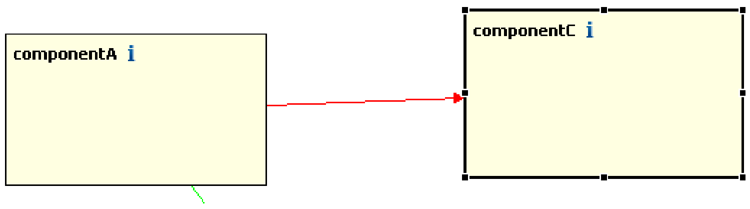
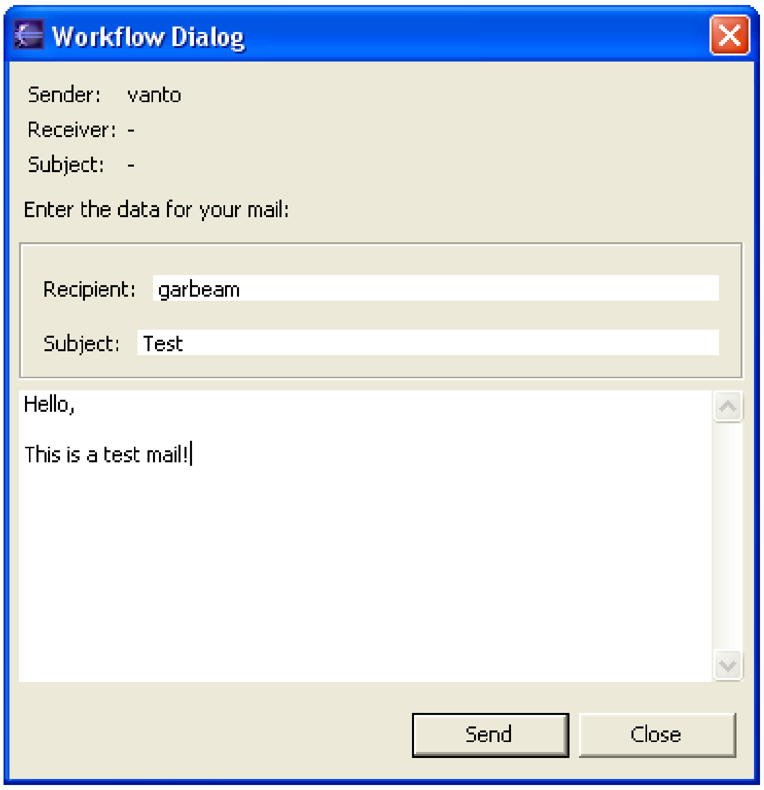


Figure 6.17: Exclusion edge

6.6 Message-related

6.6.1 Writing a mail

In the menu of the Workflow View select ”new mail”. A Workflow window opens where you can enter your message, the subject and the recipient of the message. Send the message by pressing the ”Send” button. Pressing the ”Cancel” button will close the window without sending your message (see 6.18).

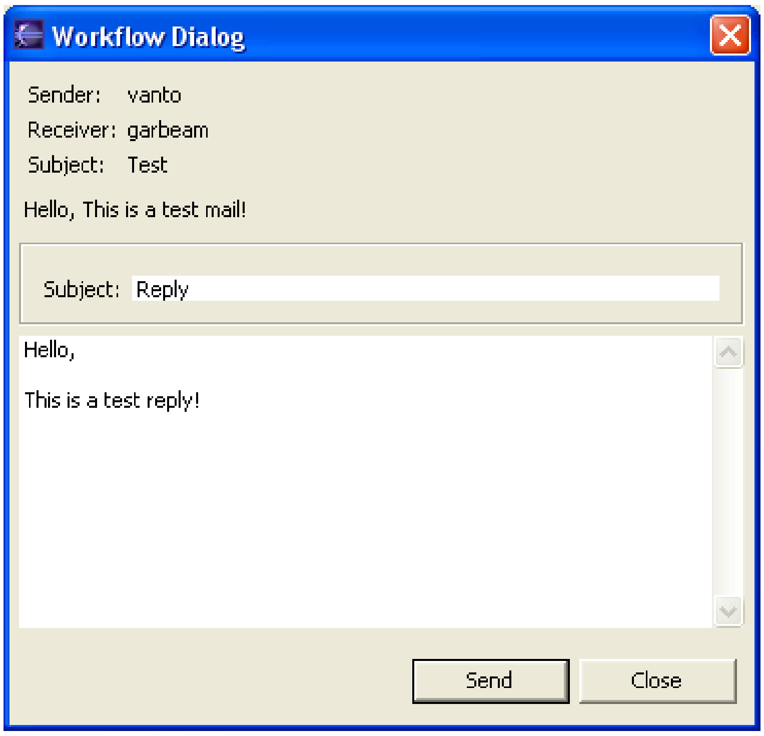


Sender: vanto
Receiver: -
Subject: -
Enter the data for your mail:
Recipient: garbeam
Subject: Test
Hello,
This is a test mail!

Figure 6.18: Write a new mail

6.6.2 Answering a mail

In the Workflow View double-click on the mail you want to answer. A Workflow window opens where you can see the message text of the mail. Below you can enter the subject and the text of your reply. Send the answer by pressing the ”Send” button. Pressing the ”Cancel” button will close the window without sending your message (see 6.19).



Sender: vanto
Receiver: garbeam
Subject: Test

Hello, This is a test mail!

Subject: Reply

Hello,

This is a test reply!

Figure 6.19: Answer a mail

6.6.3 Deleting a message

Right-click on the message and choose ”Delete message” in the context menu.

6.6.4 Fetching messages

Select a project. Right-click in the Workflow View or open the corresponding menu. Choose ”Fetch message”. Your new messages are being fetched and displayed in the Workflow View.

6.6.5 Suggesting an asset for a Core Group

There are two way to suggest an asset for a Core Group.

In the menu of the Workflow View select ’Suggest asset for core group’. A dialog opens where you can select the type of recipient, i.d. whether the suggestion is to be sent to a PE or a PLE. After that decision a Workflow window opens where you can enter the name of the asset and the username of the PE/PLE you want to send the message to. You can also enter an additional comment you want the PE/PLE to read. Send the message by pressing the ”Send” button. Pressing the ”Cancel” button will close the window without sending your message (see 6.20).

In the Role View, select the asset you want to suggest and right-click on it. The context menu opens where you choose the option ’Suggest asset for core group’. A dialog opens where you can select the type of recipient, i.d. whether the suggestion is to be sent to a PE or a PLE. After that decision a Workflow window opens where you can enter the name of the asset and the username of the PE/PLE you want to send the message to. You can also enter an additional comment you want the PE/PLE to read. Send the message by pressing the ”Send” button. Pressing the ”Cancel” button will close the window without sending your message (see 6.20).

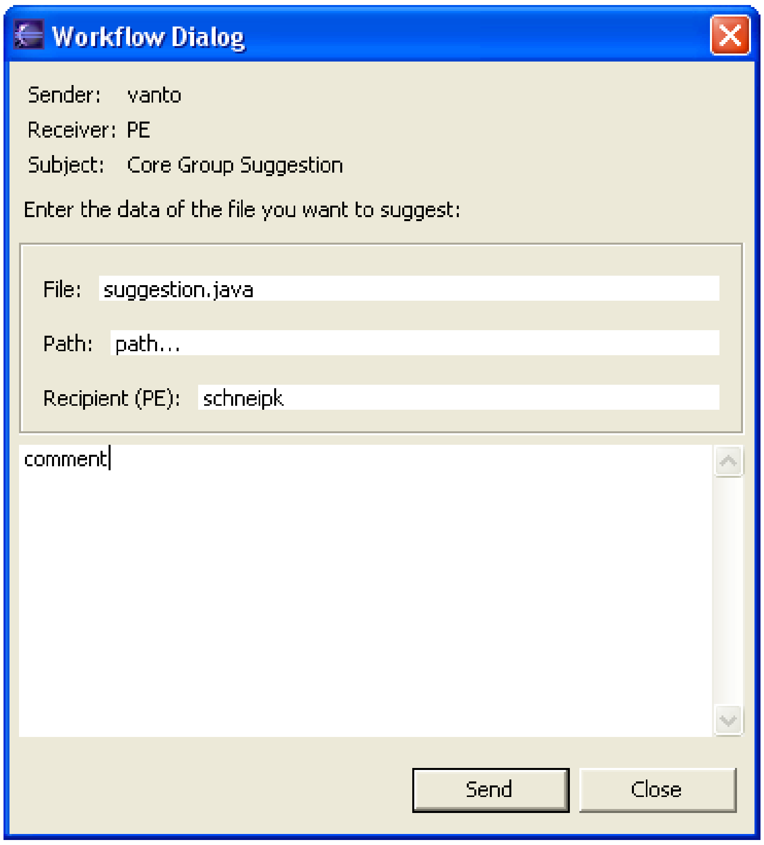
6.6.6 Dealing with a Core Group suggestion

In the Workflow View double-click on the Core Group suggestion message. A Workflow window opens where you can see the message of the programmer (see 6.21) or PE (see 6.22). Below you can select whether you agree to the suggestion or whether you decline it. You can also enter an additional comment if you like. Send the message by pressing the ”Send” button. Pressing the ”Cancel” button will close the window without sending your message.

Note: The file will not be automatically uploaded and committed. The PLE has to do this manually.

6.7 Exporting through GXL

Select the productline, product, component or variant you want to export. In the context menu of the object select ”export” (see 6.23). A wizard opens (see 6.24) where you can enter the path and name of the gxl-file. Alternatively you can enter the data through the ”browse” dialog. To start the export, press the OK button. You are able to see the status of your export in the wizard which will close after the export is finished. If you decide not to do the export, simply press the ”cancel” button.



Sender: vanto
Receiver: PE
Subject: Core Group Suggestion
Enter the data of the file you want to suggest:
File: suggestion.java
Path: path...
Recipient (PE): schneipk
comment

Figure 6.20: Suggesting a file for a Core Group

6.8 Creating a Workflow

Workflows are triggered by specific rules which you can find in the ruleset.drl file. Here you can find some existing rules as an example.

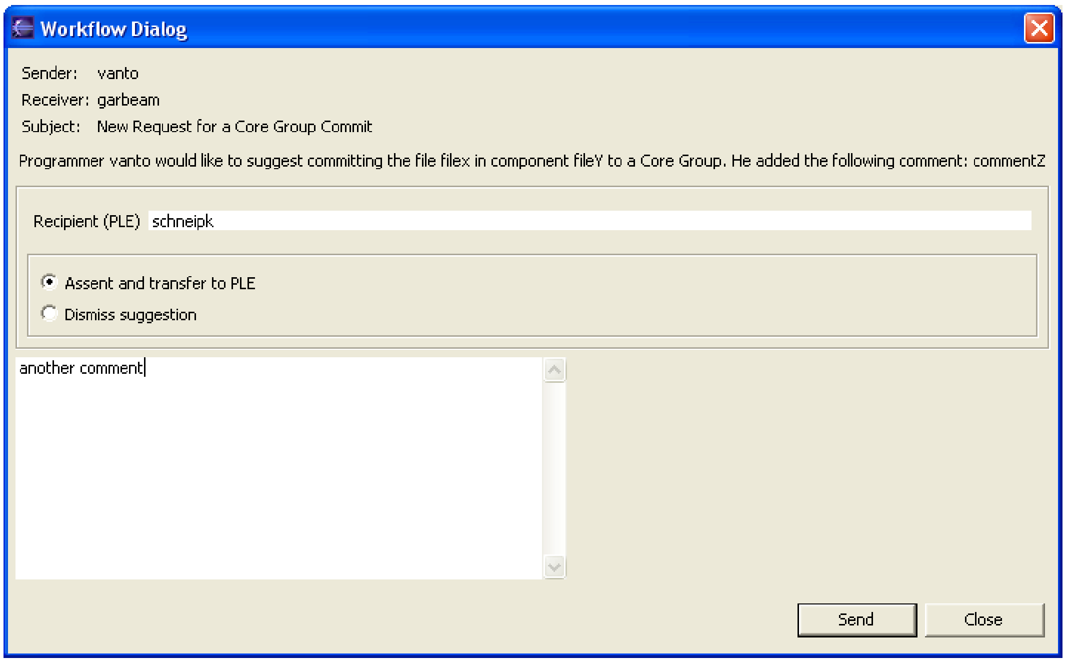
A rule consists of the following parts:

- name
- parameters
- conditions
- one consequence

Rules are a mixture of XML and Java.

This is what a sample rule looks like:

<rule name = ”Name of the rule”>



Sender: vanto
Receiver: garbeam
Subject: New Request for a Core Group Commit

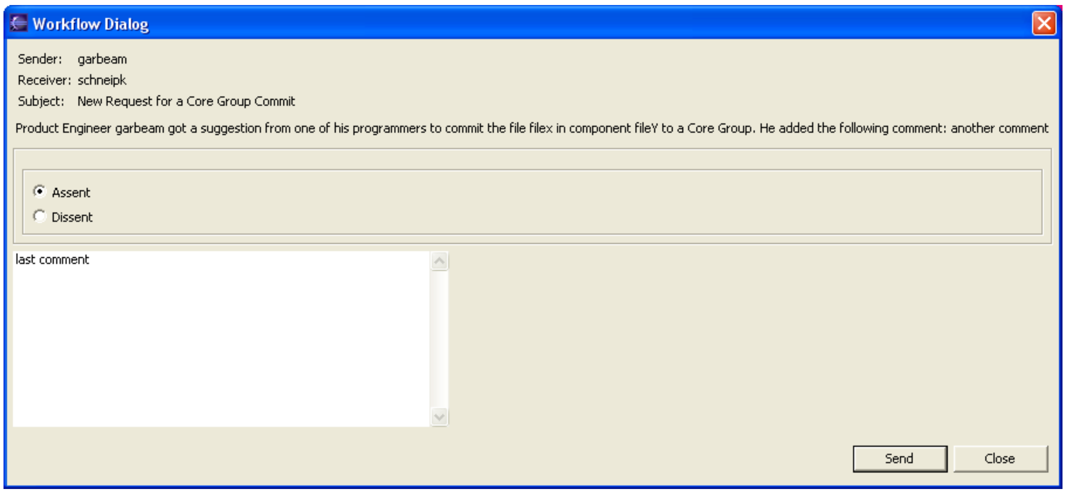
Programmer vanto would like to suggest committing the file filex in component fileY to a Core Group. He added the following comment: commentZ

Recipient (PLE) schneipk

[✓ Assent and transfer to PLE]
Dismiss suggestion

another comment

Figure 6.21: Dealing with a Core Group suggestion - PE



Sender: garbeam
Receiver: schneipk
Subject: New Request for a Core Group Commit

Product Engineer garbeam got a suggestion from one of his programmers to commit the file filex in component fileY to a Core Group. He added the following comment: another comment

[✓ Assent]
Dissent

last comment

Figure 6.22: Dealing with a Core Group suggestion - PLE

```
<parameter identifier = ”spy”>

    <java:class> kobold.common.data.RPCSpy </java:class>

</parameter>
<java:condition> spy.getMethodName().equals(”login”) </java:condition>
```

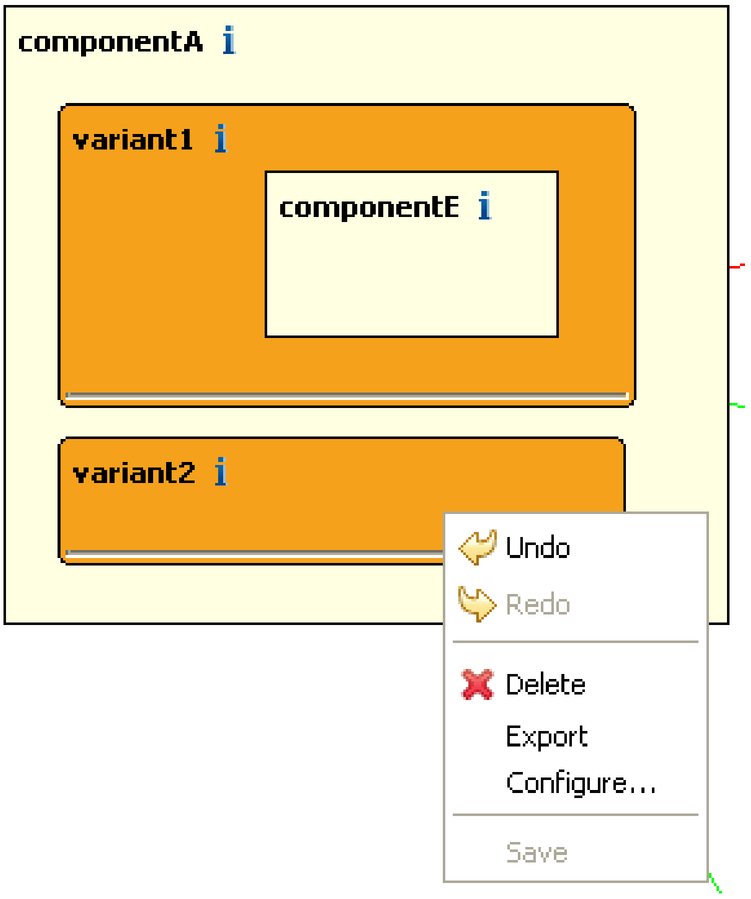


Figure 6.23: Context menu of the Architecture View

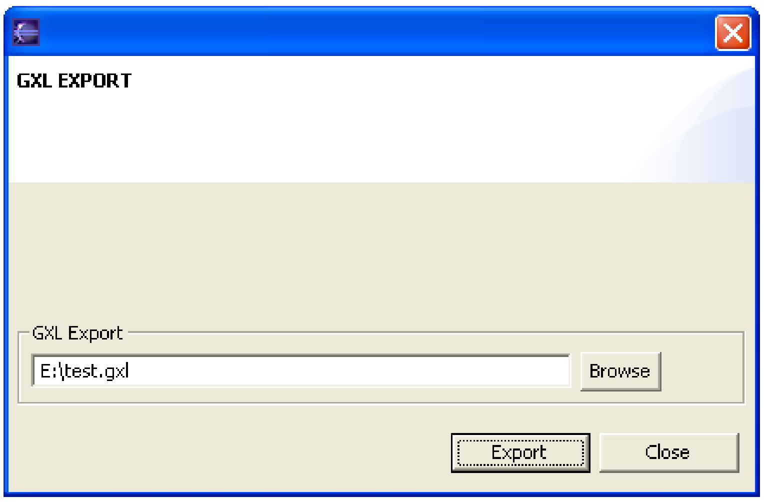


Figure 6.24: Export wizard

<java:consequence> System.out.println(”Rule fired.”); </java:consequence>

</rule>

The rule will only be fired when its conditions are true. If there are several conditions, all of them have to be true. Conditions and consequence are written in Java.

For your own rules, the rule’s parameter will always be an RPCSpy object of the package kobold.common.data. Such an object is created whenever an action is being executed by the server. These are in detail (using their actual method names):

- addUser
- getAllUsers
- removeUser
- updateUserPassword
- updateUserFullName
- getProductlineNames
- getProductline
- updateProductline
- updateProduct
- updateComponent

The parameter RPCSpy will contain the methodName which you can get with the ”getMethodName()” method. This returns one of the above Strings. An RPCSpy object also contains a vector of arguments that were transmitted to the server for the execution. You have access to this vector through the ”getArguments()” method.

The ruleset.drl file contains a sample rule for adding a user. Whenever a new user is added, every other user is notified.

In the last chapter you find the source code of the classes you need to work with for writing your own rules. In order to have an up-to-date appendix, check out the complete Kobold module. Then enter the following command in the console within the directory of the user manual:

```
perl appendix.pl >appendix.tex
```

After that, rebuild the usermanual.pdf.

6.9 Generating a metainfo document

In the role tree right-click on the object (productline, product, component, variant, release) you want the metainfo document about and select ’generate document...’. A ’Save as’ dialog opens where you can select the parent folder into which the pdf-file will be saved. Confirm by pressing the ’OK’ button and the pdf-file is created.

6.10 Configuring an asset

In the role tree right-click on the object (productline, product, component, variant, release) you want to configure and select ’configure asset...’. The corresponding ’Asset Configuration’ dialog opens where you can make your changes (see 6.12 and 6.15).

6.11 User-related

6.11.1 Creating a user

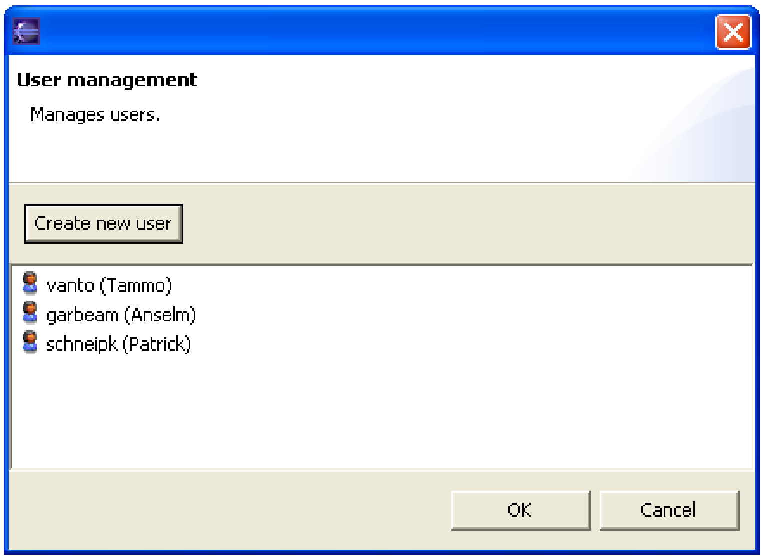
Right-click in the role tree. In the context menu choose ’create new user’. The User Manager (see 6.25) opens where you see a list of all existing users. Push the ’create new user’ button. In the opening dialog you can enter the username, real name and the initial password of the user (see 6.26). Pressing the ’OK’ button creates the new user and you can see the new user in the list of the User Manager.

6.11.2 Deleting a user

Right-click in the role tree and select ’remove user’. A dialog opens with a list of all existing users (see 6.27). Simply select the check-box next to the user you want to delete and confirm with ’OK’. The user has been deleted.

6.11.3 Changing ones password

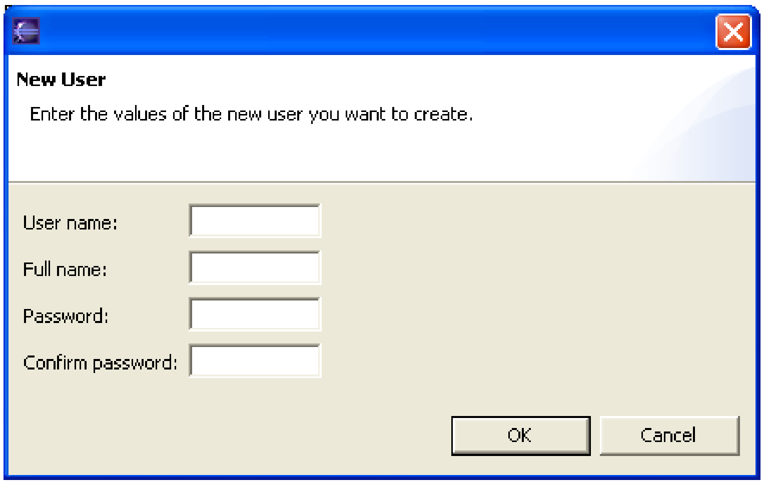
Right-click in the role tree and select ’change password’. A dialog opens where you can enter your new password (see 6.28). Confirm the password and press ’OK’. Your password has been changed.



User management

Manages users.

Figure 6.25: Create a new user - step 1



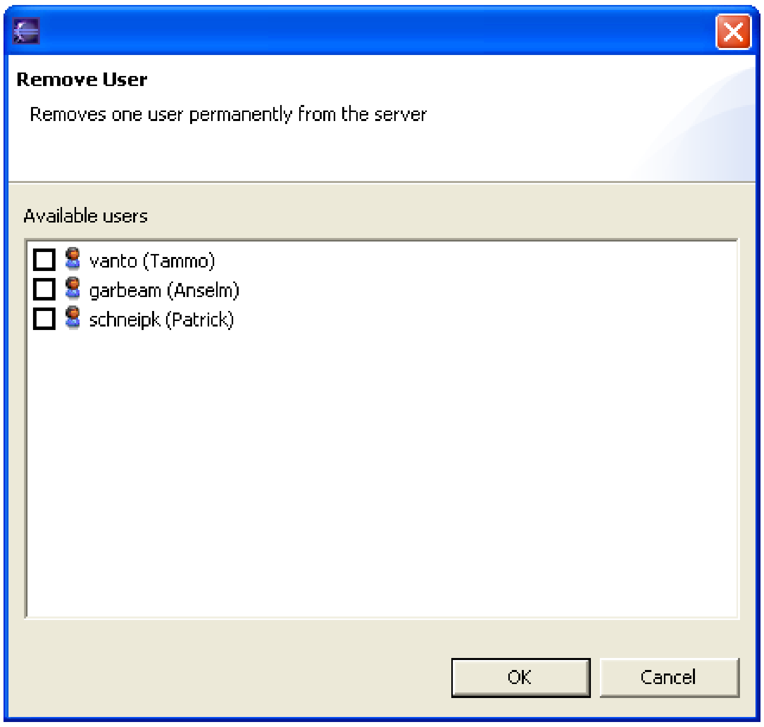
New User

Enter the values of the new user you want to create.

Figure 6.26: Create a new user - step 2

6.11.4 Updating ones full name

Right-click in the role tree and select ’update full name’. A dialog opens where you can enter your new full name (see 6.29). Confirm the changes with your password and press ’OK’. Your full name has been changed.



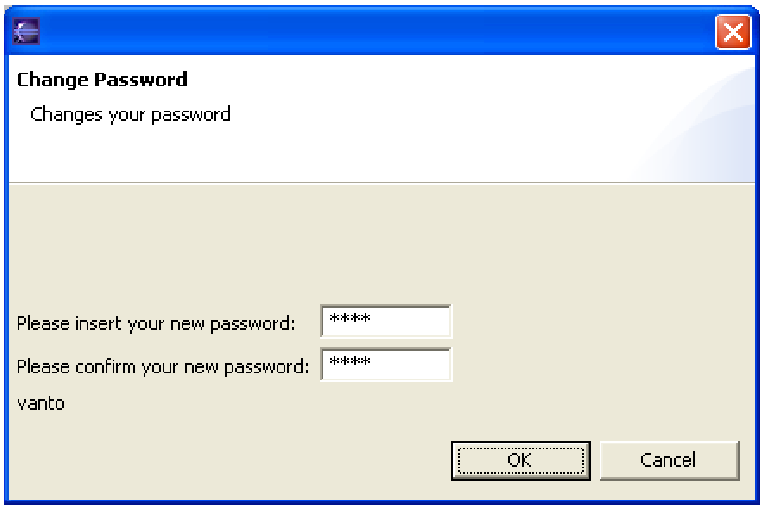
Remove User

Removes one user permanently from the server

Available users

- [ ] vanto (Tammo)
- [ ] garbeam (Anselm)
- [ ] schneipk (Patrick)

Figure 6.27: Remove a user



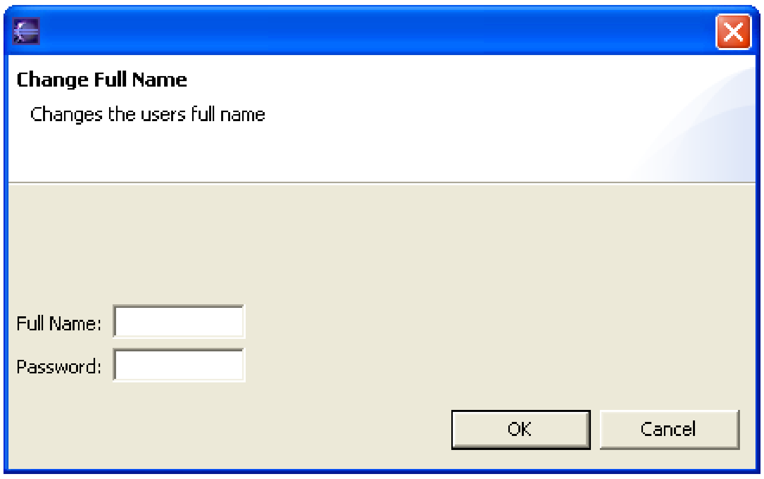
Change Password

Changes your password

Please insert your new password:	****
Please confirm your new password:	****

vanto

Figure 6.28: Change your password


Change Full Name

Changes the users full name

Full Name:

Password:

Figure 6.29: Update your full name

6.12 SAT-related

6.12.1 Creating a product line

Enter the command 'newpl'. You are asked to enter the name of the new product line and its meta data. Once you confirmed your entry, the product line is created.

6.12.2 Deleting a product line

Enter the command 'rempl'. You are then asked to enter the name of the product line you want to remove. Once you confirmed your entry, the product line is deleted.

6.12.3 Upgrading an existing user to PLE status

Enter the command 'assignple'. You are asked to enter the name of the person you want to upgrade as PLE. Finally, enter the name of the product line you want to assign the new PLE to. Once you confirmed your entry, the user has PLE rights.

6.12.4 Removing PLE rights

Enter the command 'unassignple'. You are asked to enter the name of the user who shall no longer have PLE rights. Also, enter the name of the product line you want to remove the PLE from. Once you confirmed your entry, the user is no longer PLE of the entered product line.

6.12.5 Getting a list of all existing commands

Enter the command ’help’ and you get a list of all the commands of this tool and their meanings.

6.12.6 Getting information about SAT’s version and licence

Enter the command ’about’ and you get information about the programm’s version and licence.

6.12.7 Changing the URL

Enter the command ’seturl’. Enter a new url to change the current url.

6.12.8 Changing the password

Enter the command ’setpasswd’ and then a new password in order to change the password that is used to access the current server.

6.12.9 Creating a new user

Enter the command ’newuser’. Enter the name, username and password of the new user and confirm.

6.12.10 Deleting a user

Enter the command ’remuser’. Enter the username of the user in order to remove him/her from the server.

6.12.11 Getting a list of all productlines

Enter the command ’pllist’ in order to get a list of all existing productlines.

6.12.12 Getting a list of all ples of a productline

Enter the command ’plelist’ and then the name of the productline in order to get a list of the assigned ples.

6.12.13 Getting a list of all registered users

Enter the command ’userlist’ in order to get a list of all existing users.

6.12.14 Exiting the tool

Enter the command ”exit”.

7 Source Code

7.1 IKoboldServer

```
/*
 * Copyright (c) 2003 - 2004 Necati Aydin, Armin Cont,
 * Bettina Druckenmueller, Anselm Garbe, Michael Grosse,
 * Tammo van Lessen,  Martin Plies, Oliver Rendgen, Patrick Schneider
 *
 * Permission is hereby granted, free of charge, to any person obtaining
 * a copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
 * DEALINGS IN THE SOFTWARE.
 *
 * $Id: IKoboldServer.java,v 1.16 2004/08/02 14:00:55 garbeam Exp $
 *
 */

package kobold.common.controller;

import java.net.URL;
import java.util.Vector;

import kobold.common.data.AbstractKoboldMessage;
import kobold.common.data.Productline;
import kobold.common.data.User;
import kobold.common.data.UserContext;

/**
 * This class acts as an interface between kobold clients and a
 * kobold server.
 */
public interface IKoboldServer {

public static final String NO_RESULT = "NO_RESULT";

/**
 * Login handler.
 * @param url the server url.
 * @param userName the username.
 * @param password the plain text password.
 * @return UserContext, if the userName and password
 *    is valid.
 */
    public UserContext login(URL url, String userName, String password);

/**
 * Logout handler.
 * Invalidates the given user context.
 * @param userContext the user context.
 */
    public void logout(UserContext userContext);

/**
 * Adds an new user to the server.
 * @param userContext the user context of the valid creator of the
 *    new user (if the new user is a P, than the userContext
 *    must be at least a PE).
 * @param userName the user name.
 * @param password the password.
 * @param fullName the full name.
 */
public void addUser(UserContext userContext,
String userName,
String password,
String fullName);

    /**
     * Get list of all users.
     * @param userContext
     */
    public Vector getAllUsers(UserContext userContext);

/**
     * Applies modifications to the specified user fullname.
 * @param userContext the user context
 * @param user the user name
 * @param password the decrypted user password verification
     */
    public void updateUserFullName(UserContext userContext,
                User user, String password);

/**
     * Applies modifications to the specified user password.
 * @param userContext the user context
 * @param user the user name
 * @param oldPassword the old password
 * @param newPassword the new password
     */
    public void updateUserPassword(UserContext userContext,
            User user, String oldPassword,
            String newPassword);

/**
     * Removes the specified user.
     * @param userContext the user context.
     * @param user the user to remove.
     */
    public void removeUser(UserContext userContext, User user);

    /**
     * Fetches a productline by its name.
     * @param userContext the user context.
     * @param id the id of the productline.
     * @return the product line.
     */
    public Productline getProductline(UserContext userContext, String id);

    /**
     * Fetches all product line names.
     * @param userContext the user context.
     * @return {@see java.util.List} of the productline names.
     */
    public Vector getProductlineNames(UserContext userContext);
    /**
 * Applies modifications to the given Productline.
 * If you make changes to a specific product or component,
 * use the specific method instead.
 * @param userContext the user context.
 * @param productline the productline.
 */
public void updateProductline(UserContext userContext,
  Productline id);

/**
 * Sends a KoboldMessage or WorkflowMessage.
 *
 * @param userContext the user context.
 * @param koboldMessage the message.
 */
public void sendMessage(UserContext userContext,
AbstractKoboldMessage koboldMessage);


/**
 * Fetches a single KoboldMessage. Should be put to a queue.
 * Note: to remove the message from Servers message queue,
 * it has to be invalidated using invalidateMessage!
 *
 * @param userContext the user context.
 */
public AbstractKoboldMessage fetchMessage(UserContext userContext);


/**
 * Invalidates the specified message. This method will remove the message
 * from Servers message queue.
 *
 * @param userContext the user context.
 * @param koboldMessage the message.
 */
public void invalidateMessage(UserContext userContext,
  AbstractKoboldMessage koboldMessage);
}
```

7.2 AbstractKoboldMessage

```
/*
 * Copyright (c) 2003 - 2004 Necati Aydin, Armin Cont,
 * Bettina Druckenmueller, Anselm Garbe, Michael Grosse,
 * Tammo van Lessen,  Martin Plies, Oliver Rendgen, Patrick Schneider
 *
 * Permission is hereby granted, free of charge, to any person obtaining
 * a copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
 * DEALINGS IN THE SOFTWARE.
 *
 * $Id: AbstractKoboldMessage.java,v 1.5 2004/08/01 11:53:11 garbeam Exp $
 *
 */
package kobold.common.data;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.dom4j.DocumentHelper;
import org.dom4j.Element;

/**
 * @author garbeam
 */
public abstract class AbstractKoboldMessage implements ISerializable {
```

```
private static final Log logger = LogFactory.getLog(AbstractKoboldMessage.class)
private DateFormat dateFormat = new SimpleDateFormat("yyMMddHHmmssSZ");

/** this field must be changed if you subclass this class */

public static final String STATE_UN_FETCHED = "UN_FETCHED";
public static final String STATE_FETCHED = "FETCHED";
public static final String STATE_INVALID = "INVALID";

public static final String PRIORITY_HIGH = "high";
public static final String PRIORITY_NORMAL = "normal";
public static final String PRIORITY_LOW = "low";

private String sender;
private String receiver;
private String messageText;
private Date date = new Date();
private String priority = PRIORITY_NORMAL;
private String subject;
private String id;
private String state = STATE_UN_FETCHED;

/**
 * Creates a new Kobold Message.
 * Use this constructor to use a seperate id pool for the given type.
 * @param idtype
 */
protected AbstractKoboldMessage(String idtype)
{
id = IdManager.nextId(idtype);
}

/**
 * Returns the state.
 * @return state.
 */
public String getState() {
return state;
}

/**
 * @return
 */
public Date getDate() {
return date;
}

/**
 * @return
 */
public String getId() {
return id;
}

public abstract String getType();

/**
 * @return
 */
public String getMessageText() {
return messageText;
}

/**
 * @return
 */
public String getPriority() {
return priority;
}

/**
 * @return
 */
public String getReceiver() {
return receiver;
}

/**
 * @return
 */
public String getSender() {
return sender;
}

/**
 * @return
 */
public String getSubject() {
return subject;
}
```

```
/**
 * @param string
 */
public void setDate(Date date) {
this.date = date;
}

/**
 * @param i
 */
protected void setId(String id) {
this.id = id;
}

/**
 * @param string
 */
public void setMessageText(String string) {
messageText = string;
}

/**
 * @param string
 */
public void setPriority(String string) {
priority = string;
}

/**
 * @param string
 */
public void setReceiver(String string) {
receiver = string;
}

/**
 * @param string
 */
public void setSender(String string) {
sender = string;
}

/**
 * @param string
 */
public void setSubject(String string) {
subject = string;
}

/**
 * Serializes this object to an xml element and adds it to the given root.
 * @param root
 */
public Element serialize() {
Element xmsg = DocumentHelper.createElement("message");
xmsg.addAttribute("type", getType());
xmsg.addAttribute("id", id);
xmsg.addAttribute("priority", priority);
xmsg.addAttribute("state", state);

if (sender != null) {
    xmsg.addElement("sender").setText(sender);
}

if (receiver != null) {
    xmsg.addElement("receiver").setText(receiver);
}

xmsg.addElement("date").setText(date.getTime() + "");

if (subject != null) {
    xmsg.addElement("subject").setText(subject);
        }

if (messageText != null) {
    xmsg.addCDATA(messageText);
}

return xmsg;
}

/**
 * Deserializes message
 */
public void deserialize(Element data) {
id = data.attributeValue("id");
priority = data.attributeValue("priority");
state = data.attributeValue("state");
```

```
sender = data.elementTextTrim("sender");
receiver = data.elementTextTrim("receiver");

// fall back on parse error
date = new Date();
String d = data.elementTextTrim("date");
if (d != null) {
date.setTime(Long.parseLong(d));
}

subject = data.elementTextTrim("subject");

messageText = data.getTextTrim();
}


/**
 * @see java.lang.Object#equals(java.lang.Object)
 */
public boolean equals(Object obj)
{
if (!(obj instanceof AbstractKoboldMessage))
return false;
return ((AbstractKoboldMessage)obj).getId().equals(getId());
}

/**
 * @see java.lang.Object#hashCode()
 */
public int hashCode()
{
return getId().hashCode();
}

/**
 * @see java.lang.Object#toString()
 */
public String toString()
{
StringBuffer sb = new StringBuffer(getClass().getName());
sb.append("\n\t[id:       " + getId() + "]\n");
sb.append("\t[state:   "  + getState() + "]\n");
sb.append("\t[sender:   " + getSender() + "]\n");
sb.append("\t[receiver: " + getReceiver() + "]\n");
sb.append("\t[subject:  " + getSubject() + "]\n");
```

```
return sb.toString();
}

/**
 * Factorymethod to create a Kobold-/Workflow-instance from an dom4j element.
 * Checks the type attribute to select the right class, returns null if type
 * attribute is not set or has wrong data.
 *
 * @param el
 * @return
 */
public static AbstractKoboldMessage createMessage(Element el)
{
String type = el.attributeValue("type");
if (type == null)
return null;

if (type.equals(KoboldMessage.TYPE)) {
return new KoboldMessage(el);
}
else if (type.equals(WorkflowMessage.TYPE)) {
return new WorkflowMessage(el);
}
else return null;
}

/**
 * Sets the state.
 * @param state the state.
 */
public void setState(String state) {
this.state = state;
}
}
```

7.3 Asset

```
/*
 * Copyright (c) 2003 - 2004 Necati Aydin, Armin Cont,
 * Bettina Druckenmueller, Anselm Garbe, Michael Grosse,
 * Tammo van Lessen,  Martin Plies, Oliver Rendgen, Patrick Schneider
 *
 * Permission is hereby granted, free of charge, to any person obtaining
 * a copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
 * DEALINGS IN THE SOFTWARE.
 *
 * $Id: Asset.java,v 1.6 2004/08/05 09:12:58 vanto Exp $
 *
 */
package kobold.common.data;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import kobold.common.io.RepositoryDescriptor;

import org.dom4j.DocumentHelper;
import org.dom4j.Element;

/**
 * Base class for architectural elements which are managed by the Kobold Server.
 * Implements the {@see kobold.common.data.ISerializable} class for client-serve
 * interchange.
 */
public class Asset implements ISerializable {

public static final String COMPONENT = "component";
public static final String PRODUCT = "product";
public static final String PRODUCT_LINE = "productline";

// resource (file or directory name) of this asset
private String resource = null;
```

```
// name of this asset
private String name = null;
// id of this asset
private String id = null;
// type of this asset
private String type = null;
// repository location of this asset
private RepositoryDescriptor repositoryDescriptor = null;
// maintainers of this asset
private List maintainer = new ArrayList();
// parent asset
private Asset parent = null;

/**
 * Base constructor for server side assets.
 * @param parent of this asset, if <code>null</code> this
 *    asset is the root.
 * @param type the type of this asset, must be a value of the
 *    static final members of this class.
 * @param id a unique id of this asset.
 * @param name the name
 * @param resource the file or directory name (no PATH!)
 * @param repositoryDescriptor
 */
public Asset(Asset parent, String type, String name, String resource,
     RepositoryDescriptor repositoryDescriptor)
{
this.parent = parent;
this.type = type;
this.name = name;
this.resource = resource;
this.repositoryDescriptor = repositoryDescriptor;

this.id = IdManager.nextId(name);
}


public Asset(Asset parent)
{
    this.parent = parent;
}

/**
 * DOM constructor for deserialization.
 */
```

```
// public Asset(Asset parent, Element element) {
// this.parent = parent;
// deserialize(element);
// }

/**
 * Serializes this asset.
 */
public Element serialize() {
Element element = DocumentHelper.createElement(type);
element.addAttribute("id", this.id);
element.addAttribute("name", this.name);
element.addAttribute("resource", this.resource);
if (parent != null) {
element.addAttribute("parent-id", parent.getId());
}
element.add(repositoryDescriptor.serialize());
Element maintainerElements = element.addElement("maintainers");
for (Iterator iterator = maintainer.iterator(); iterator.hasNext(); ) {
User user = (User) iterator.next();
maintainerElements.add(user.serialize());
}
return element;
}

/**
 * Deserializes this asset.
 */
public void deserialize(Element element) {
this.type = element.getName();
this.id = element.attributeValue("id");
this.name = element.attributeValue("name");
this.resource = element.attributeValue("resource");
this.repositoryDescriptor =
new RepositoryDescriptor(element.element("repository-descriptor"));
Element maintainerElements = element.element("maintainers");
for (Iterator iterator = maintainerElements.elementIterator("user");
     iterator.hasNext(); )
{
Element elem = (Element) iterator.next();
maintainer.add(new User(elem));
}
}

/**
```

```
 * Sets the id of this asset.
 * @param id
 */
public void setId(String id) {
    this.id = id;
}
/**
 * Returns the id of this asset.
 */
public String getId() {
return id;
}

/**
 * Returns a list of all maintainer.
 */
public List getMaintainers() {
return maintainer;
}

/**
 * Adds new maintainer.
 * @param user the maintainer.
 */
public void addMaintainer(User user) {
maintainer.add(user);
}

/**
 * Removes maintainer.
 * @param user the maintainer.
 */
public void removeMaintainer(User user) {
maintainer.remove(user);
}

/**
 * Returns the name of this asset.
 */
public String getName() {
return name;
}

/**
 * Returns the parent asset.
 */
public Asset getParent() {
return parent;
}

/**
 * Sets the parent.
 * @parent parent the parent asset.
 */
public void setParent(Asset parent) {
this.parent = parent;
}

/**
 * Returns the repositoryDescriptor.
 */
public RepositoryDescriptor getRepositoryDescriptor() {
return repositoryDescriptor;
}

/**
 * Returns the type.
 */
public String getType() {
return type;
}

    public String getResource() {
        return resource;
    }

    public void setResource(String resource) {
        this.resource = resource;
    }
}
```

7.4 Component

```
/*
 * Copyright (c) 2003 - 2004 Necati Aydin, Armin Cont,
 * Bettina Druckenmueller, Anselm Garbe, Michael Grosse,
 * Tammo van Lessen,  Martin Plies, Oliver Rendgen, Patrick Schneider
 *
 * Permission is hereby granted, free of charge, to any person obtaining
 * a copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
 * DEALINGS IN THE SOFTWARE.
 *
 * $Id: Component.java,v 1.5 2004/08/03 16:46:35 garbeam Exp $
 *
 */

package kobold.common.data;

import kobold.common.io.RepositoryDescriptor;

import org.dom4j.Element;

/**
* Represents a server-side component. If the parent is a productline this class
* represents a server-side coreasset. Used for client-server interchange.
 */
public class Component extends Asset {

/**
 * Basic constructor.
 * @param parent the parent asset, e.g. a product or productline.
 * @param name the name of this component.
 * @param resource the resource
 * @param repositoryDescriptor the repository descriptor of this
 *    component.
 */
public Component(Asset parent, String name, String resource,
                 RepositoryDescriptor repositoryDescriptor)
{
```

```
super(parent, Asset.COMPONENT, name, resource, repositoryDescriptor);
}

/**
 * DOM constructor.
 * @param parent the parent of this component.
 * @param element the DOM element representing this component.
 */
public Component (Asset parent, Element element) {
super(parent);
super.deserialize(element);
}

/**
 * Serializes this component.
 */
public Element serialize() {
Element element = super.serialize();
return element;
}
}
```

7.5 IdManager

```
/*
 * Copyright (c) 2003 - 2004 Necati Aydin, Armin Cont,
 * Bettina Druckenmueller, Anselm Garbe, Michael Grosse,
 * Tammo van Lessen,  Martin Plies, Oliver Rendgen, Patrick Schneider
 *
 * Permission is hereby granted, free of charge, to any person obtaining
 * a copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
 * DEALINGS IN THE SOFTWARE.
 *
 * $Id: IdManager.java,v 1.6 2004/07/25 23:17:48 vanto Exp $
 *
 */
package kobold.common.data;

import java.io.IOException;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;

import kobold.common.KoboldCommonsPlugin;

import org.apache.commons.id.IdentifierUtils;
import org.apache.commons.id.uuid.NodeManager;
import org.apache.commons.id.uuid.UUID;
import org.apache.commons.id.uuid.clock.Clock;
import org.apache.commons.id.uuid.state.Node;
import org.apache.commons.id.uuid.state.State;
import org.apache.commons.id.uuid.state.StateHelper;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

/**
 * Provides an id manager using the IETF UUID standard.
 *
 * @author Tammo van Lessen
 */
public class IdManager
{
    private static final Log logger = LogFactory.getLog(IdManager.class);

public static String nextId(String idtype) {
String id = ((UUID)IdentifierUtils.UUID_VERSION_ONE_GENERATOR.nextIdentifier()).
//logger.debug("new id for "+idtype+": "+id);
return id;
}

public static class NodeManagerImpl implements NodeManager {

    /** Reference to the State implementation to use for loading and storing */
    private State nodeState;
    /** The current array index for the Node in use. */
    private int currentNodeIndex = 0;
    /** Flag indicating the node state has been initialized. */
    private boolean isInit = false;
    /** Set that references all instances. */
    private Set nodesSet;
    /** Array of the Nodes */
    private Node[] allNodes;
    /** UUID timestamp of last call to State.store */
    private long lastUUIDTimeStored = 0;

    public NodeManagerImpl() {
        // tell common-plugin my instance.
        KoboldCommonsPlugin.getDefault().setNodeManager(this);
    }

    /*** Initialization */
    public void init() {
        nodeState = new EclipseUUIDState();
        nodeState.load();
        nodesSet = nodeState.getNodes();
        Iterator it = nodesSet.iterator();
        allNodes = new Node[nodesSet.size()];
        int i = 0;
        while (it.hasNext()) {
            allNodes[i++] = (Node) it.next();
        }
        isInit = true;
    }

    public void store() {
        try {
                nodeState.store(nodesSet);
            } catch (IOException e) {
                // nothing to do here.
            }
    }

        /* (non-Javadoc)
         * @see org.apache.commons.id.uuid.NodeManager#currentNode()
         */
        public Node currentNode()
        {
            if (!isInit) {
```

```
            init();
        }
        // See if we need to store state information.
        if ((lastUUIDTimeStored + nodeState.getSynchInterval()) > (findMaxTi
            try {
                nodeState.store(nodesSet);
            } catch (IOException ioe) {
                //@TODO add listener and send notify
            }
        }
        return allNodes[currentNodeIndex];
    }

    /* (non-Javadoc)
     * @see org.apache.commons.id.uuid.NodeManager#nextAvailableNode()
     */
    public Node nextAvailableNode()
    {
        if (!isInit) {
            init();
        }
        currentNodeIndex++;
        if (currentNodeIndex >= allNodes.length) {
            currentNodeIndex = 0;
        }
        return currentNode();
    }

    /**
     * <p>Returns the maximum uuid timestamp generated from all <code>Node</
     *
     * @return maximum uuid timestamp generated from all <code>Node</code>s.
     */
    private long findMaxTimestamp() {
        if (!isInit) {
            init();
        }
        long max = 0;
        for (int i = 0; i < allNodes.length; i++) {
            if (allNodes[i] != null && allNodes[i].getLastTimestamp() > max)
                max = allNodes[i].getLastTimestamp();
            }
        }
        return max;
    }
```

```
        public void lockNode(Node arg0)
        {
            // not implemented
        }
        public void releaseNode(Node arg0)
        {
            // not implemented
        }

}

private static class EclipseUUIDState implements State
{
    private HashSet nodes = new HashSet(1);
    private Node node = null;

    /**
     * @see org.apache.commons.id.uuid.state.State#load()
     */
    public void load() throws IllegalStateException
    {
        String nodeId = KoboldCommonsPlugin.getDefault().getPluginPreferences().
        long lastTime = KoboldCommonsPlugin.getDefault().getPluginPreferences().
        short seq = (short)KoboldCommonsPlugin.getDefault().getPluginPreferences

        node = null;
        if ("".equals(nodeId)) {
            logger.debug("no node found, creating a new one");
            node = new Node(StateHelper.randomNodeIdentifier());
        } else {
            logger.debug("node found. using node "+ nodeId +"");
            node = new Node(StateHelper.decodeMACAddress(nodeId), lastTime, seq)
        }

        nodes.add(node);
    }

    /**
     * @see org.apache.commons.id.uuid.state.State#getNodes()
     */
    public Set getNodes()
    {
        return nodes;
    }
    /**
     * @see org.apache.commons.id.uuid.state.State#store(java.util.Set)
     */
    public void store(Set arg0) throws IOException
    {
        KoboldCommonsPlugin.getDefault().getPluginPreferences().setValue("uuid-n
        KoboldCommonsPlugin.getDefault().getPluginPreferences().setValue("uuid-l
        KoboldCommonsPlugin.getDefault().getPluginPreferences().setValue("uuid-c
        KoboldCommonsPlugin.getDefault().savePluginPreferences();
        logger.debug("uuid node stored.");
    }

    /**
     * @see org.apache.commons.id.uuid.state.State#store(java.util.Set, long)
     */
    public void store(Set arg0, long arg1)
    {
       logger.error("store(Set, int) not implemented!");
    }

    /**
     * @see org.apache.commons.id.uuid.state.State#getSynchInterval()
     */
    public long getSynchInterval()
    {
        return 3000;//Long.MAX_VALUE;
    }

}

}
```

7.6 KoboldMessage

```
/*
 * Copyright (c) 2003 - 2004 Necati Aydin, Armin Cont,
 * Bettina Druckenmueller, Anselm Garbe, Michael Grosse,
 * Tammo van Lessen,  Martin Plies, Oliver Rendgen, Patrick Schneider
 *
 * Permission is hereby granted, free of charge, to any person obtaining
 * a copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
 * DEALINGS IN THE SOFTWARE.
 *
 * $Id: KoboldMessage.java,v 1.12 2004/05/18 18:47:33 vanto Exp $
 *
 */
package kobold.common.data;

import org.dom4j.Element;

/**
 * @author Tammo
 */
public class KoboldMessage extends AbstractKoboldMessage
{
    public static final String TYPE = "kobold";

/**
 * Creates a new Kobold Message with a new unique id. (type = kmesg).
 */
public KoboldMessage()
{
super("kmesg");
}

/**
 * Unmarshals a Kobold Message.
 * @param data
 */
public KoboldMessage(Element data)
{
this();
super.deserialize(data);
}

    /**
     * @see kobold.common.data.AbstractKoboldMessage#getType()
     */
    public String getType()
    {
    return KoboldMessage.TYPE;
    }

}
```

7.7 Product

```
/*
 * Copyright (c) 2003 - 2004 Necati Aydin, Armin Cont,
 * Bettina Druckenmueller, Anselm Garbe, Michael Grosse,
 * Tammo van Lessen,  Martin Plies, Oliver Rendgen, Patrick Schneider
 *
 * Permission is hereby granted, free of charge, to any person obtaining
 * a copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
 * DEALINGS IN THE SOFTWARE.
 *
 * $Id: Product.java,v 1.13 2004/08/03 16:46:35 garbeam Exp $
 *
 */
```

```
package kobold.common.data;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import kobold.common.io.RepositoryDescriptor;
import kobold.common.data.Productline;

import org.dom4j.Element;

/**
 * Represents a server side product. Used for client-server interchange.
 */
public class Product extends Asset {

private List components = new ArrayList();

/**
 * Basic constructor.
 * @param productline the parent productline.
 * @param name the name of this product.
 * @param resource the file or directory name
 * @param repositoryDescriptor the repository descriptor of this product.
 */
public Product (Productline productline, String name, String resource,
               RepositoryDescriptor repositoryDescriptor) {
super(productline, Asset.PRODUCT, name, resource, repositoryDescriptor);
}

/**
 * DOM constructor.
 * @param productline the parent productline.
 * @param the DOM element representing this asset.
 */
public Product (Productline productline, Element element) {
    super(productline);
deserialize(element);
}

/**
 * Returns all components of this product.
 */
public List getComponents() {
return components;
```

```
}

/**
 * Adds new component to this product.
 * @param component the component.
 */
public void addComponent(Component component) {
components.add(component);
}

/**
 * Removes an existing component from this product.
 * @param component the component.
 */
public void removeComponent(Component component) {
components.remove(component);
}

/**
 * Serializes this product.
 */
public Element serialize() {
Element element = super.serialize();

Element compElements = element.addElement("components");
for (Iterator iterator = components.iterator(); iterator.hasNext(); ) {
Component component = (Component) iterator.next();
compElements.add(component.serialize());
}

return element;
}

/**
 * Deserializes this product. It’s asserted that super deserialization
 * is already finished.
 * @param element the DOM element representing this product.
 */
public void deserialize(Element element) {
super.deserialize(element);
    Element compElements = element.element("components");

for (Iterator iterator = compElements.elementIterator(Asset.COMPONENT);
 iterator.hasNext(); )
{
```

```
Element elem = (Element) iterator.next();
components.add(new Component(this, elem));
}
}
}
```

7.8 Productline

```
/*
 * Copyright (c) 2003 - 2004 Necati Aydin, Armin Cont,
 * Bettina Druckenmueller, Anselm Garbe, Michael Grosse,
 * Tammo van Lessen,  Martin Plies, Oliver Rendgen, Patrick Schneider
 *
 * Permission is hereby granted, free of charge, to any person obtaining
 * a copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
 * DEALINGS IN THE SOFTWARE.
 *
 * $Id: Productline.java,v 1.20 2004/08/03 16:46:35 garbeam Exp $
 *
 */
package kobold.common.data;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import org.dom4j.Element;

import kobold.common.io.RepositoryDescriptor;


/**
 * This class represents a server side productline. Productlines consist of
 * several products, they can be assigned Users (Productline engineers) who
 * have the right to modify it and they encapsulate the necessary information
 * about the repository where the actual products are stored.
 *
 * They are created (and removed) by the corresponding administration methods
 * and stored directly on the Kobold server.
 *
 * @see kobold.server.controller.SecureKoboldWebServer
 * @see kobold.common.data.Product
 * @see kobold.common.io.RepositoryDescriptor
 */
public class Productline extends Asset {

private Map coreassets = new HashMap();
private Map products = new HashMap();

/**
 * Base constructor for productlines.
 * @param name the name of this productline.
 * @param resource the file or directory name.
 * @param repositoryDescriptor containing the necessary information about
 *        this productlines repository
 */
public Productline(String name, String resource, RepositoryDescriptor repository
super(null, Asset.PRODUCT_LINE, name, resource, repositoryDescriptor);
}

/**
 * DOM constructor for server-side productlines.
 * @param element the DOM element representing this productline.
 */
public Productline(Element element) {
super(null);
deserialize(element);
}

/**
 * Adds a new product to this productline. Please note that each registered
```

```
     * product needs to have its own (unique) name. Adding of a prodct with a
     * name that has already been registered will be refused.
     *
 * @param product the product to add.
     * @return true, if the passed productline could be added successfully,
     *         false otherwise
 */
public boolean addProduct(Product product) {
Object o = products.put(product.getName(), product);

if (o != null){
            // a product with the same name as the one to add already exists
            // => undo the change and signal error
            products.put(product.getName(), o);
            return false;
}
        else{
         product.setParent(this);
            return true;
        }
}

/**
 * Gets a product by its name.
 * @param productName the product name.
     * @return the product with the specified name or null if no product with
     *         that name exists
 */
public Product getProduct(String productName) {
    return (Product)products.get(productName);
}

/**
 * Gets a coreasset by its name.
 * @param coreAssetName the coreasset.
     * @return the coreasset with the specified name or null if no coreasset wit
     *         that name exists
 */
public Component getCoreAsset(String coreAssetName) {
    return (Component)products.get(coreAssetName);
}

/**
 * Gets all products.
     * @return the products
 */
public List getProducts() {
    return new ArrayList(products.values());
}

/**
 * Gets all coreassets.
     * @return the coreassets
 */
public List getCoreAssets() {
    return new ArrayList(coreassets.values());
}

/**
 * Removes the passed product from this productline if its reagistered.
 * @param product the product to remove.
     * @return the removed Product if it was part of this productline or null
     *         if not
 */
public Product removeProduct(Product product) {
Product ret = (Product) products.remove(product.getName());

        if (ret != null){
         ret.setParent(null);
        }

        return ret;
}

/**
 * Adds new core asset to this productline. Please note that each registered
     * coreasset needs to have its own (unique) name. Adding of a ca with a
     * name that has already been registered will be refused.
     *
 * @param coreasset the coreasset to add.
     * @return true if the passed coreasset could be registered successfully,
     *         false otherwise
 */
public boolean addCoreAsset(Component coreasset) {
        Object o = coreassets.put(coreasset.getName(), coreasset);

        if (o != null){
            coreassets.put(coreasset.getName(), o);
            return false;
        }
        else{
         coreasset.setParent(this);
            return true;
        }
}

/**
 * Removes a coreasset from this productline, if it is registered.
 * @param coreasset the coreassset to remove.
     * @return the removed coreasset if it existed as part of this productline
     *         or null if not
 */
public Component removeCoreAsset(Component coreasset) {
Component ret = (Component) coreassets.remove(coreasset.getName());

        if (ret != null){
         coreasset.setParent(null);
        }

        return ret;
}

/**
 * Serializes this productline.
 */
public Element serialize() {
Element element = super.serialize();

Element coreassetElements = element.addElement("coreassets");
for (Iterator iterator = coreassets.values().iterator(); iterator.hasNext(); ) {
Component component = (Component) iterator.next();
coreassetElements.add(component.serialize());
}

Element productElements = element.addElement("products");
for (Iterator iterator = products.values().iterator(); iterator.hasNext(); ) {
Product product = (Product) iterator.next();
productElements.add(product.serialize());
}

return element;
}

/**
 * Deserializes this productline. It’s asserted that super deserialization
 * is already finished.
 * @param element the DOM element representing this productline.
 */
public void deserialize(Element element) {
super.deserialize(element);
    Element coreassetElements = element.element("coreassets");
for (Iterator iterator = coreassetElements.elementIterator(Asset.COMPONENT);
 iterator.hasNext(); )
{
Element elem = (Element) iterator.next();
Component component = new Component(this, elem);
coreassets.put(component.getName(), component);
}

Element productElements = element.element("products");
for (Iterator iterator = productElements.elementIterator(Asset.PRODUCT);
 iterator.hasNext(); )
{
Element elem = (Element) iterator.next();
Product product = new Product(this, elem);
products.put(product.getName(), product);
}
}
}
```

7.9 RPCSpy

```
/*
 * Copyright (c) 2003 - 2004 Necati Aydin, Armin Cont,
 * Bettina Druckenmueller, Anselm Garbe, Michael Grosse,
 * Tammo van Lessen,  Martin Plies, Oliver Rendgen, Patrick Schneider
 *
 * Permission is hereby granted, free of charge, to any person obtaining
 * a copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
 * DEALINGS IN THE SOFTWARE.
 *
 * $Id: RPCSpy.java,v 1.1 2004/06/23 13:27:26 garbeam Exp $
 *
 */

package kobold.common.data;

import java.util.Vector;

/**
 * Container for RPC call sniffing.
 */
public class RPCSpy {

private String methodName = null;
private Vector arguments = null;

/**
 * Clones all RPC stuff.
 * @param methodName the method which has been invoked.
 * @param arguments the argument container.
 */
public RPCSpy(String methodName, Vector arguments) {
this.methodName = methodName;
this.arguments = arguments;
}

/**
 * @return Returns the arguments.
 */
public Vector getArguments() {
return arguments;
}
/**
 * @return Returns the methodName.
 */
public String getMethodName() {
return methodName;
}
}
```

7.10 User

```
/*
 * Copyright (c) 2003 - 2004 Necati Aydin, Armin Cont,
 * Bettina Druckenmueller, Anselm Garbe, Michael Grosse,
 * Tammo van Lessen,  Martin Plies, Oliver Rendgen, Patrick Schneider
 *
 * Permission is hereby granted, free of charge, to any person obtaining
 * a copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
 * DEALINGS IN THE SOFTWARE.
 *
 * $Id: User.java,v 1.14 2004/07/07 15:40:32 garbeam Exp $
 *
 */
package kobold.common.data;


import org.dom4j.DocumentHelper;
import org.dom4j.Element;


/**
 * Provides a user representation on the client side.
 *
 * Implements ISerializable to make the tranport through XMLRPC possible.
 *
 * @author Tammo
 */
public class User implements ISerializable
{
    private String username;
    private String fullname;


    public User(String username, String fullname)
    {
        this.username = username;
        this.fullname = fullname;
    }

    public User(Element element) {
     deserialize(element);
    }

    /**
     * @param fullname The fullname to set.
     */
    public void setFullname(String fullname)
    {
        this.fullname = fullname;
    }

    /**
     * @return Returns the fullname.
     */
    public String getFullname()
    {
        return fullname;
    }

    /**
     * @param username The username to set.
     */
    public void setUsername(String username)
    {
        this.username = username;
    }

    /**
     * @return Returns the username.
     */
    public String getUsername()
    {
        return username;
    }

    /**
     * @see kobold.common.data.ISerializable#serialize()
     */
    public Element serialize()
    {
        Element user = DocumentHelper.createElement("user");
        if (username != null) {
            user.addAttribute("username", username);
        }

        if (fullname != null) {
            user.addAttribute("fullname", fullname);
        }

        return user;
    }

    /**
     * @see kobold.common.data.ISerializable#deserialize(org.dom4j.Element)
     */
    public void deserialize(Element element)
    {
       username = element.attributeValue("username");
       fullname = element.attributeValue("fullname");
    }

    public boolean equals(Object obj)
    {
        if (obj instanceof User) {
            return getUsername().equals(((User)obj).getUsername());
        }
        return false;
    }

    public int hashCode()
    {
        return getUsername().hashCode();
    }
}
```

7.11 UserContext

```
/*
 * Copyright (c) 2004 Armin Cont, Anselm R. Garbe, Bettina Druckenmueller,
 *                    Martin Plies, Michael Grosse, Necati Aydin,
 *                    Oliver Rendgen, Patrick Schneider, Tammo van Lessen
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
 * copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
 * in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
 * THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
 * ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
 * OTHER DEALINGS IN THE SOFTWARE.
 *
 * $Id: UserContext.java,v 1.10 2004/08/02 14:00:55 garbeam Exp $
 */

package kobold.common.data;

import java.net.URL;

import org.dom4j.DocumentHelper;
import org.dom4j.Element;

/**
 * Base class for user context information (session info).
 * This class provides information about the current user context
 * which is the session info.
 *
 * @author garbeam
 */
public class UserContext implements ISerializable {
// members
private String userName;
private String sessionId;
private URL serverUrl = null;

    /**
     * DOM constructor for user contexts.
 * @param element the DOM element represendting this user context.
 */
public UserContext(Element element) {
deserialize(element);
}


    /**
     * Default constructor, which provides basic info
     * about the user context.
     * This class could be used to determine more information
     * about the specific user context from the Kobold server,
     * e.g. roles, repository access data, etc.
     *
     * @param username username of the users (like a Unix username).
     * @param sessionId the current session Id of the user context.
     */
    public UserContext(String userName, String sessionId)
    {
        this.userName = userName;
        this.sessionId = sessionId;
    }

    /**
     * @returns the username of this user context.
     */
    public String getUserName() {
        return this.userName;
    }

    /**
     * @returns the session id of this user context.
     */
    public String getSessionId() {
        return this.sessionId;
    }

/**
 * Sets sessionId.
 */
```

```
public void setSessionId(String sessionId) {
this.sessionId = sessionId;
}

/**
 * Serializes this object.
 *
 * @return DOM Element representing this object.
 */
public Element serialize() {
Element userContext = DocumentHelper.createElement("usercontext");
userContext.addElement("username").addText(this.userName);
userContext.addElement("session-id").addText(this.sessionId);

return userContext;
}

/**
 * Deserializes this object.
 * @param element the DOM element representing this object.
 */
public void deserialize(Element element) {
this.userName = element.elementText("username");
this.sessionId = element.elementText("session-id");
}

    public URL getServerUrl() {
        return serverUrl;
    }

    public void setServerUrl(URL serverUrl) {
        this.serverUrl = serverUrl;
    }
}
```

7.12 WorkflowItem

```
/*
 * Copyright (c) 2003 - 2004 Necati Aydin, Armin Cont,
 * Bettina Druckenmueller, Anselm Garbe, Michael Grosse,
 * Tammo van Lessen,  Martin Plies, Oliver Rendgen, Patrick Schneider
 *
 * Permission is hereby granted, free of charge, to any person obtaining
 * a copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
 * DEALINGS IN THE SOFTWARE.
 *
 * $Id: WorkflowItem.java,v 1.7 2004/05/19 16:08:34 martinplies Exp $
 *
 */
package kobold.common.data;

import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;

import org.dom4j.DocumentHelper;
import org.dom4j.Element;

/**
 * @author pliesmn
 *
 * A WorkflowItem is a Checkbutton, a Radiobutton or a  single-Line-TextWindow
 * The data, to display a WorkFlowItem is stored here.
 *
 */
public class WorkflowItem {

public final static String TEXT = "text";
public final static String CHECK = "check";
public final static String RADIO = "radio";
public final static String CONTAINER = "container";

private String description;
private String value; // The Name of the Checkbutton or Textwindow or the value
private String type;
private List children = new LinkedList();

public WorkflowItem(Element data)
{
deserialize(data);
}

/**
 * Creates a new WorkflowItem object.
 *
 * There are 4 different kinds of workflow items:
 *  <ul>
 *  <li>type = TEXT: value = text in inputarea, description, no children<li>
 *  <li>type = CHECK: value = "true"|"false", description, no children<li>
 *  <li>type = RADIO: value = "true"|"false", description, must be added to a
 *  CONTAINER item, no children<li>
 *  <li>type = CONTAINER: value = undefined|null, description, RADIO items as ch
 *  </ul>
 *
 * @param nameValue
 * @param description
 * @param type
 */
public WorkflowItem(String value, String description, String type) {
this.value = value;
this.description = description;
this.type = type;
}


public void addChild(WorkflowItem control)
{
if (!type.equals(WorkflowItem.CONTAINER))
throw new IllegalArgumentException("illegal add");
children.add(control);
}


public void addChildren(WorkflowItem[] control)
{
if (!type.equals(WorkflowItem.CONTAINER))
throw new IllegalArgumentException("illegal add");
for (int itemNr =0; itemNr <  control.length; itemNr++)
  children.add(control[itemNr]);
```

```
}

public WorkflowItem[] getChildren()
{
return (WorkflowItem[])children.toArray(new WorkflowItem[0]);
}

public String getType() {
return type;
}

public String getValue(){
return value;
}

public String getDescription() {
return this.description;
}

public Element serialize()
{
Element element = DocumentHelper.createElement("control");
element.addElement("type").setText(type);
element.addElement("description").setText(description);
element.addElement("value").setText(value);

if (type.equals(WorkflowItem.CONTAINER)) {
Element childrenEl = element.addElement("children");
Iterator it = children.iterator();
while (it.hasNext()) {
WorkflowItem wfi = (WorkflowItem)it.next();
childrenEl.add(wfi.serialize());
}
}
return element;
}

protected void deserialize(Element data)
{
type = data.elementTextTrim("type");
value = data.elementTextTrim("value");
description = data.elementTextTrim("description");

Element childrenEl = data.element("children");
if (type.equals(WorkflowItem.CONTAINER) && childrenEl != null) {
```

```
children.clear();
Iterator it = childrenEl.elementIterator("control");
while (it.hasNext()) {
Element control = (Element)it.next();
children.add(new WorkflowItem(control));
}
}
}

public void setValue(String value)
{
this.value = value;
}

}
```

7.13 WorkflowMessage

```
/*
 * Copyright (c) 2003 - 2004 Necati Aydin, Armin Cont,
 * Bettina Druckenmueller, Anselm Garbe, Michael Grosse,
 * Tammo van Lessen,  Martin Plies, Oliver Rendgen, Patrick Schneider
 *
 * Permission is hereby granted, free of charge, to any person obtaining
 * a copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
 * DEALINGS IN THE SOFTWARE.
 *
 * $Id: WorkflowMessage.java,v 1.25 2004/06/23 13:27:26 garbeam Exp $
 *
 */
package kobold.common.data;
import java.util.*;

import org.dom4j.Element;

/**

 * @author garbeam
 * @author vanto
 */
public class WorkflowMessage extends AbstractKoboldMessage {

public static final String TYPE = "workflow";
public static final String DATA_VALUE_TRUE = "TRUE";
public static final String DATA_VALUE_FALSE= "FALSE";
private String workflowType = new String();
private String comment = "";
private Set parents = new HashSet();
private List controlItems = new LinkedList(); // ArrayList performs much better
private HashMap workflowData = new HashMap();
private int step;


/**
 * Creates a Workflow Message in the ’workflow’ id namespace
 */
public WorkflowMessage(String workflowType)
{
super(TYPE);
this.workflowType = workflowType;
}

/**
 * Creates a Workflow Message and deserializes its data from xml
 *
 * @param data a &lt;message&gt; element
 */
public WorkflowMessage(Element data)
{
super(TYPE);
    deserialize(data);
}
```

```
public void addParentId(String id)
{
parents.add(id);
}

public String[] getParentIds()
{
return (String[])parents.toArray(new String[0]);
}

public void addWorkflowControl(WorkflowItem item)
{
controlItems.add(item);
}

public WorkflowItem[] getWorkflowControls()
{
return (WorkflowItem[])controlItems.toArray(new WorkflowItem[0]);
}

/**
 * @return
 */
public String getComment() {
return comment;
}


/**
 * Sets the Workflow ID.
 * This ID describes, which Workflow should be applied by Drools

 * @return workflow id
 */
public String getWorkflowType() {
return workflowType;
}

/**
 * @param string
 */
public void setComment(String string) {
comment = string;
}
```

```
/**
 * Sets the Workflow ID.
 * This ID describes, which Workflow should be applied by Drools
 */
public void setWorkflowType(String type) {
workflowType = type;
}

/**
 * @return
 */
public Map getWorkflowData() {
return workflowData;
}

public void putWorkflowData(String key, String value) {
this.workflowData.put(key, value);
}

public String getWorkflowDate(String key) {
return (String) this.workflowData.get(key);
}

/**
 * @see kobold.common.data.KoboldMessage#deserialize(org.dom4j.Element)
 */
public void deserialize(Element data)
{
super.deserialize(data);
// TODO check why those slutty variables dont get initialized.
//parents = new HashSet();
//controlItems = new LinkedList();

workflowType = data.elementTextTrim("workflow-id");
comment = data.elementTextTrim("comment");
step = Integer.valueOf(data.elementTextTrim("step")).intValue();

Element history = data.element("history");
Iterator it = history.elementIterator("parent");

while (it.hasNext()) {
Element p = (Element)it.next();
parents.add(p.getTextTrim());
}
```

```
Element controls = data.element("controls");
it = controls.elementIterator("control");

while (it.hasNext()) {
Element c = (Element)it.next();
controlItems.add(new WorkflowItem(c));
}

Element answers = data.element("results");
it = answers.elementIterator();

while (it.hasNext()) {
Element a = (Element) it.next();
this.putWorkflowData(a.getName(), a.getTextTrim());
}

}

public String getType()
{
return WorkflowMessage.TYPE;
}

/**
 * @see kobold.common.data.KoboldMessage#serialize()
 */
public Element serialize()
{
Element el = super.serialize();
el.addElement("workflow-id").setText(workflowType);
el.addElement("comment").addCDATA(comment);
el.addElement("step").setText(String.valueOf(step));

Element hist = el.addElement("history");
Iterator it = parents.iterator();
while (it.hasNext()) {
String id = (String)it.next();
hist.addElement("parent").setText(id);
}

Element controls = el.addElement("controls");
it = controlItems.iterator();
while (it.hasNext()) {
WorkflowItem control = (WorkflowItem)it.next();
controls.add(control.serialize());
}

// store answers
Element answers = el.addElement("results");
it = workflowData.keySet().iterator();
while (it.hasNext()) {
String key = (String) it.next();
answers.addElement(key).setText((String) workflowData.get(key));
}
return el;
}

public String toString() {
StringBuffer sb = new StringBuffer();
sb.append("\t[wf-id:    " + getWorkflowType() + "]\n");
sb.append("\t[parents:  " + getParentIds().length + "]\n");
sb.append("\t[controls: " + getWorkflowControls().length + "]\n");

return super.toString() + sb.toString();
}
/**
 * @return Returns the step.
 */
public int getStep() {
return step;
}
/**
 * @param step The step to set.
 */
public void setStep(int step) {
this.step = step;
}
}
```