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 productline management system Kobold.

1.2 Background information

Short time-to-market cycles, high product quality und low costs are the key factors for success in the field of software engineering. These incompatible goals can be reached through systematic reuse throughout the development of software.

The gist of the productline approach is to systematically gather the differences and commonalities of different products and to explicitly picture these aspects in form of a software architecture. This effects all steps in the development process and in the design and strategy of products.

In order to establish an efficient productline development in the field of software engineering, you have to put this product strategy rigorously into practice.

The ideal goal of a well-configured productline is to create a product with a simple combination of core assets which are customized for the product. Therefore a productline is ideally configured if the productline architecture is sufficiently specified for all possible products. This is the task of a product(line) engineer. He specifies the architecture by defining core assets and their main relations (scoping and domain engineering). If a new product is to be added to the productline (application engineering), it has to follow this architecture. In order to achieve this, existing variants of core assets are combined with new variants und transferred to the product engineer who takes responsibility for this product. He has to make sure that the development of the product does not interfere with the architecture of the productline.

In this hierarchy it is important that procedures are performed according to fixed rules. For example, if an existing variant of a core asset is already used in a product, it mustn’t be changed by the product engineer. Changes are only allowed by the responsible core asset developer. He has to decide whether the changes are useful since his core asset is also used in other products. If he declines the changes, the product engineer has to develop a product-specific variant of the core asset. This process is very complex because several people take part in the decision. In order to enforce a productline rigorously these operations must be specified.

Kobold supports the development of productlines, simplifies the configuration management, the communication and the development and helps to model operations

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 three parts. The first part provides descriptions of each window and each menu item.

The second part is a short tutorial through which you can start working with Kobold right away.

The third part is a list of the interfaces you need in order to implement your own workflow rules (for more information see chapter 4).

2 Installation

2.1 Technical Requirements

Kobold runs on

- Windows NT/2000/XP
- Linux (Motif)
- Linux (GTK 2)
- Solaris 8 (SPARC/Motif)
- Mac OSX (Mac/Cocoa)

with the following requirements:

- 600 MHz
- 256 MB main memory
- 200 MB fixed-disk storage

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

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

2.2 Installation

To make distributing and installation easier a set of files and the necessary make shell script is located at /home/stsopra/werkbold/install_scripts (change ME) for an easy linux installation. It also includes the binaries of java, eclipse and perl (also for windows).

2.2.1 Create a distribution iso

To create a distribution iso the make.sh script is executed with the argument “distribute”. The script creates:

- the distributing directory
- the iso-file directory
- the generated iso-file

The generated iso-file is used to burn a cd-rom or can be easily mounted into the VM-ware.

2.2.2 Install Kobold

To install Kobold, the generated iso-file is needed. Mount the iso-file and change into the install-directory. The make.sh script is executed with the argument “install” and the used install path of the user. All needed files are installed to the install path and the start scripts for the server, the sat-tool and the kobold application are created and adapted to the choosen install path.

2.2.3 Adapt the PATH-variable

After the installation the user can add the shown path to his PATH-variable to start the tools without using the full path. The tcsh path is extended in the file /.cshrc with the command “setenv PATH KOBOLD_HOME:PATH”.

2.2.4 Start the tools

The tools are started with the start-scripts located in the bin directory:

- start-server.sh to start the server
- start-sat.sh to start the server administartion tool
- start-kobold.sh to get Kobold running

The start script initialize java and perl of the Kobold package and set other settings.

3 Overview of Kobold

The essential architecture of Kobold is divided in two subsystems: the Kobold Client which is a development environment for productlines and products, and the Kobold Server which administrates the users, roles and messages.

The Client is based on Eclipse and offers the user a graphical user interface through which he can administrate his productlines and products. He can view and alter the architectures in form of graphs, send messages and trigger workflows.

The Server administrates the data of each user and their responsibilities. Also it offers the Clients a central message service . If a message is sent from a Client to the Server, the Server checks the message for possible consequences which are assigned to other clients as workflows. Besides the Server administrates the paths and the access configurations of the repositories which save the data of the products and the productlines.

3.1 Architecture Objects

3.1.1 Core Asset

Core Assets are abstract modules in the productline architecture. They can contain variants.

3.2 Variant

Whenever there are more than one design possibilites, a variant is created. It allows the user to pursue both possibilites.

3.3 Release

Releases are created to save the state of a variant at a specific date.

3.3.1 Related Component

Related components are instances of releases and used in product architectures.

3.3.2 Specific Component

Specific components are components that are used in the product but not in the productline.

3.3.3 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.

3.3.4 Exclusion Edge

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

3.3.5 Meta Node

Meta nodes are used to create complex relationships between nodes. They represent the relationships AND and OR. In an AND relationship all objects are needed. The OR relationship lets you decide between the objects of the relationship. Be careful with the direction of the edges that are linked to a meta node! In the following architecture component A needs components B and C. But components B and C are independent of component A (see 4.16).

3.3.6 GXL

GXL is a graph exchange language and used to export the architectures in Kobold. For more information: http://www.gupro.de/GXL/

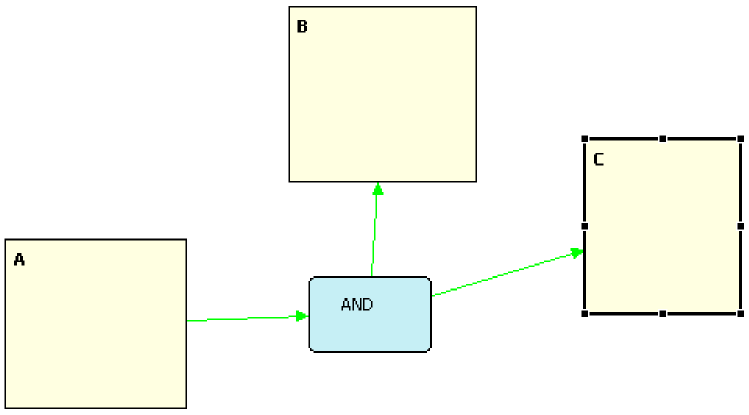


Figure 3.1: Meta node example

3.4 Roles

3.4.1 Productline Engineer (PLE)

Productline engineers have the responsibility for the productlines.

3.4.2 Product Engineer (PE)

Product engineers administrate products and core assets and are subordinates of the productline engineers.

3.4.3 Programmer (P)

Programmers are subordinates of the product engineers. They develop the software of the products and core assets.

3.5 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 productline engineer creates a basis for all the products of the productline.

3.6 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:

- add user
- get all users
- update user password
- update user full name
- get productline names
- get productline
- update productline

In Kobold workflows are triggered by either WorkflowMessage or RPCSpy objects. You can find the interfaces and implementations in chapter 6.

4 User Interface

4.1 Client

The Kobold Client is based on Eclipse. Eclipse is a kind of universal tool platform - an open extensible IDE for anything and nothing in particular. Its user interface consists of views and editors. In editors you can both see and alter the data which is displayed whereas in views you can only look at the data but not alter it.

4.1.1 Main Window

In the Main Window you can see the following editors and views (see 4.1):

- Architecture Editor
- Architecture Tree
- Navigator
- Workflow View
- Minimap

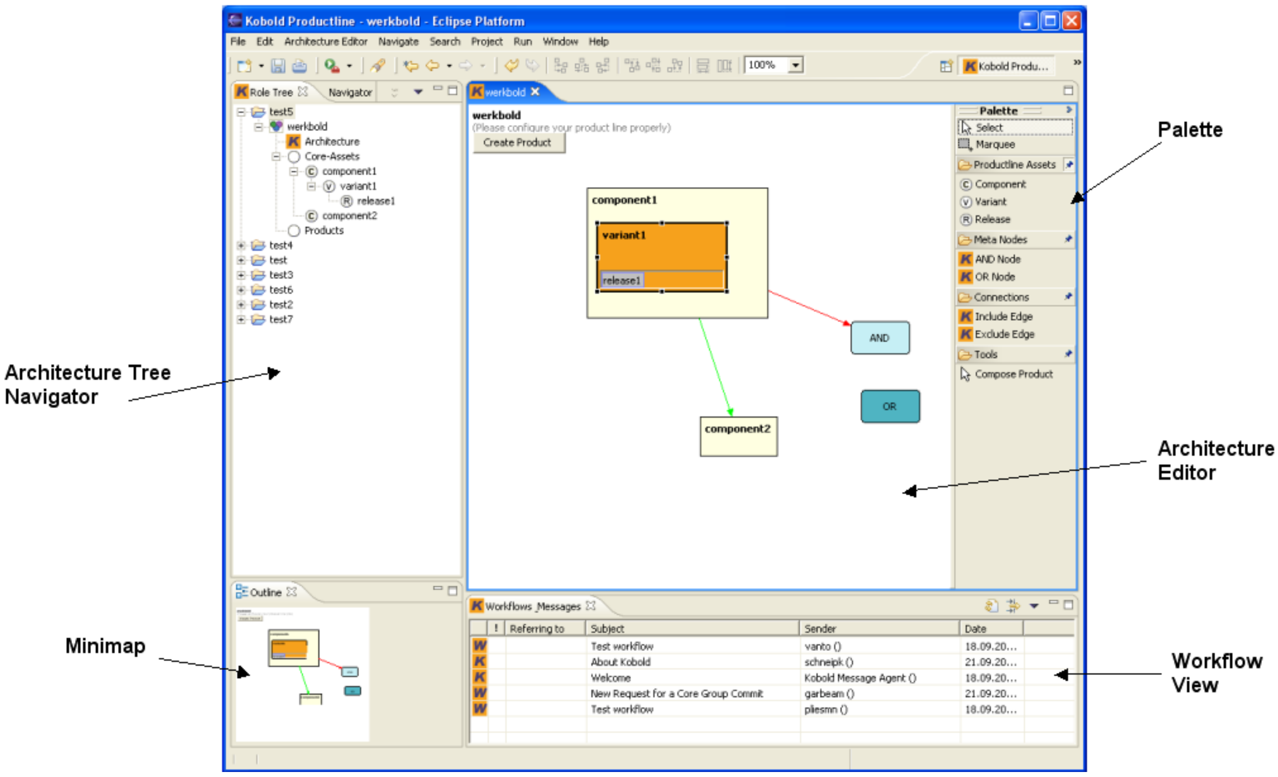
The menu offers you in addition to the Eclipse standard options the following possibilities:

- Creating a new Kobold project
- Changing Kobold properties

Creating a new project

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 4.2).

Before you proceed, you have the possibility to import a new certificate. In order to do this, press the button ’import certificate...’. A new dialog opens (see 4.3) where you can enter an alias and your new certificate. Confirm with ’OK’.



	!	Referring to	Subject	Sender	Date
W			Test workflow	vanto ()	18.09.20...
K			About Kobold	schneipk ()	21.09.20...
K			Welcome	Kobold Message Agent ()	18.09.20...
W			New Request for a Core Group Commit	garbeam ()	21.09.20...
W			Test workflow	pliesmn ()	18.09.20...

Figure 4.1: Main Window

After that you choose the productline you want to check out (see 4.4).

In the last step you have to enter the name of the project you want to create (see 4.5).

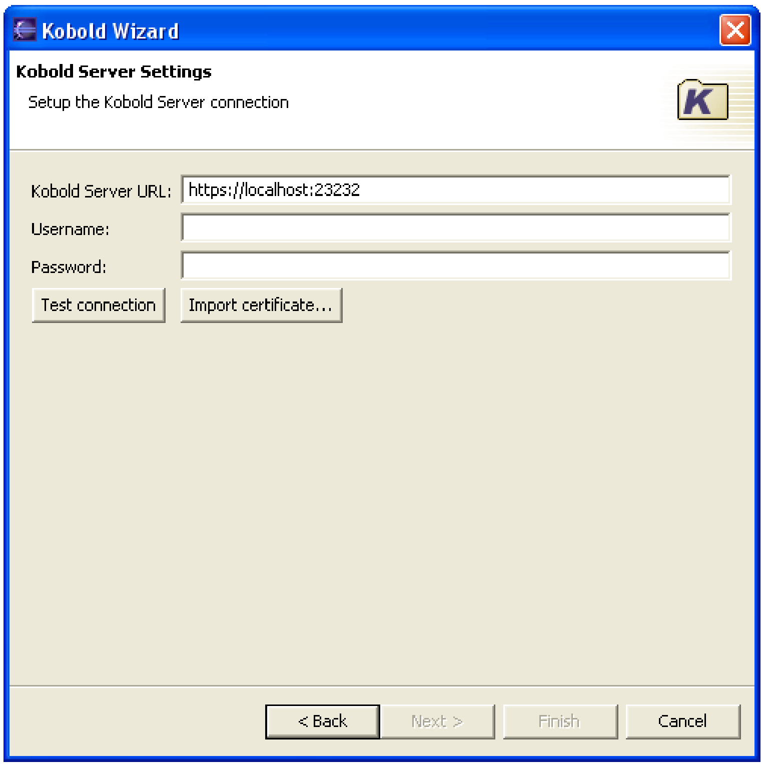
A new project has been created. You can open the different views and editors through the Window menu and 'show view'.

Setting the Kobold preferences

In the Kobold Client menu, select 'Window' and then 'Preferences'. The preferences dialog opens (see 4.6). Open the Kobold tree. In the SSL tab you can enter the properties for the Kobold Client. They resemble your SAT properties (see chapter '4.3 Server Administration Tool'). The VCM Configuration allows you to save your username, password and script for the communication with the projects' repositories.

4.1.2 Architecture Editor

The Architecture Editor (see 4.7) displays the architecture of the product, or productline respectively, that is selected in the Architecture Tree.


Kobold Wizard

Kobold Server Settings

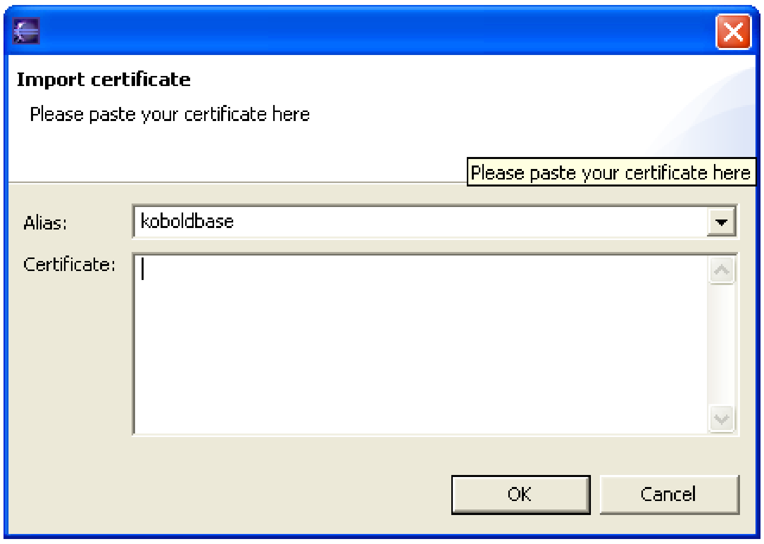
Setup the Kobold Server connection

Kobold Server URL: https://localhost:23232

Username:

Password:

Figure 4.2: Kobold wizard


Import certificate

Please paste your certificate here

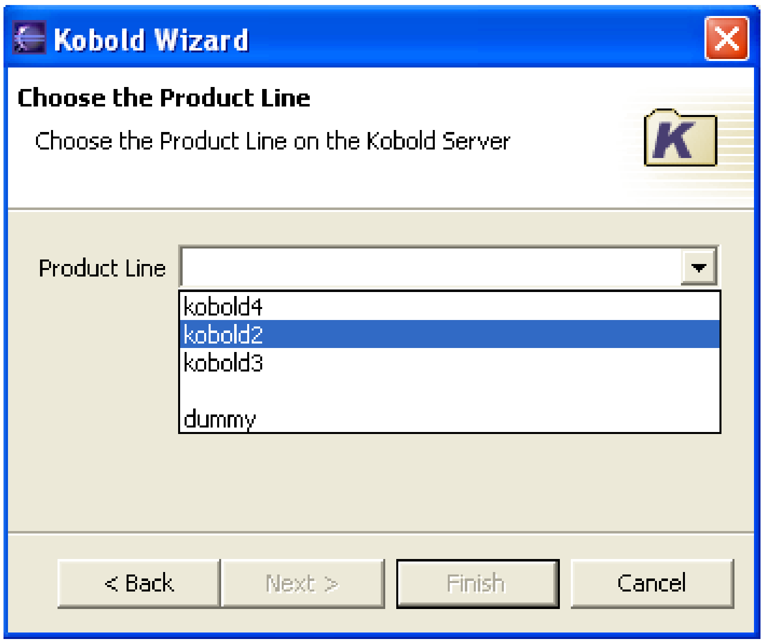
Please paste your certificate here

Alias: koboldbase

Certificate:

Figure 4.3: Importing a Certificate

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


Kobold Wizard

Choose the Product Line

Choose the Product Line on the Kobold Server

Product Line

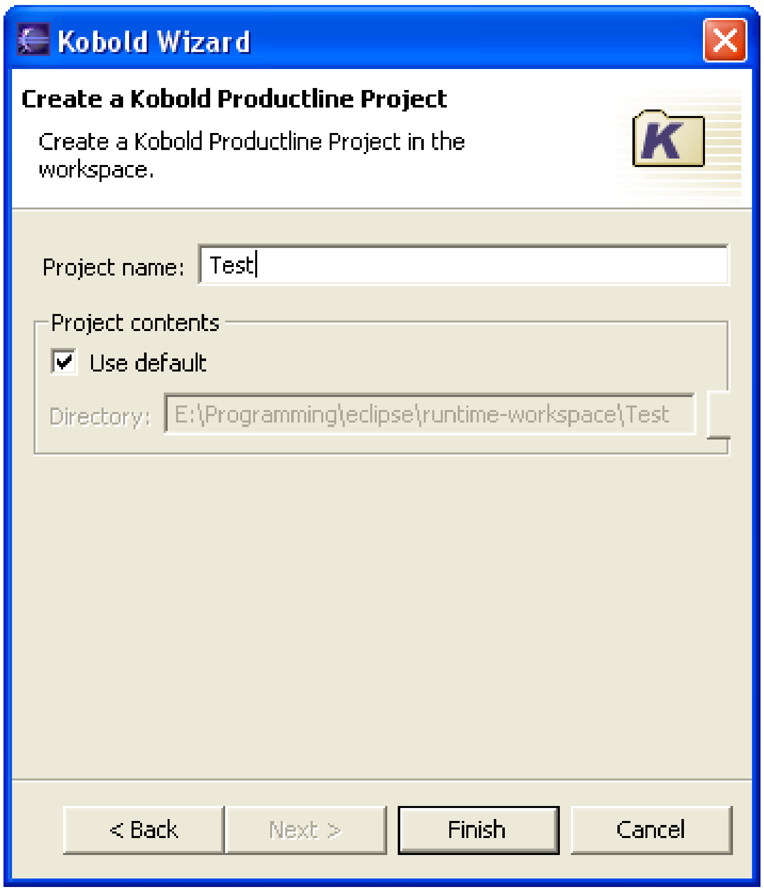
kobold4
kobold2
kobold3

dummy

Figure 4.4: Kobold wizard

The Architecture Editor offers you the following options:

- Create a core asset/component
- Configure a core asset/component
- Createa variant
- Configure a variant
- Create a release
- Configure a release
- Delete a core asset/component, variant or release
- Create a meta node
- Delete a meta node
- Create a dependency edge
- Create an exclusion edge
- Delete an edge
- Mark a product, component, variant or release ”deprecated”
- Set the maintainers of a productline, product or component
- Move an item
- Change the size of an item


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

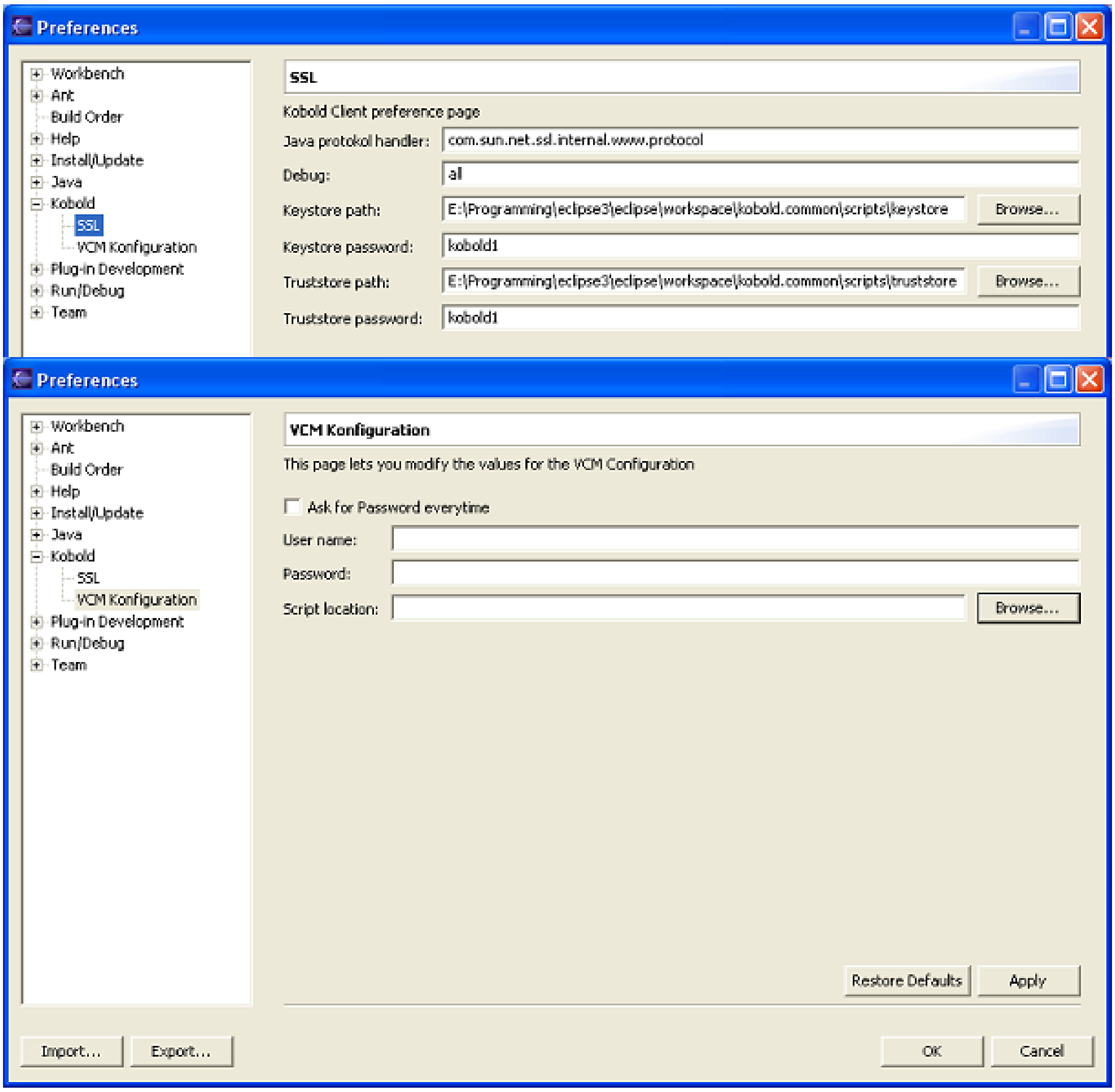
Figure 4.5: Kobold wizard

- Compose a product
- Configure a product
- Edit the repository descriptor of a product
- Export

Most of the options can be reached through the palette (see 4.8). You can find it to the right of the Architecture Editor. If offers you the tools to work with the Architecture Editor and opens up when you click on it.

Creating a Core Asset/component

In the pallete select the ”component” item. Click within a variant or top-level in the Architecture Editor. A component is inserted (see 4.9) and a dialog opens where you can enter the meta data of the component. By clicking on the border and pulling it you can simply change the size of the component.



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 4.6: Kobold preferences

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

Configuring a core asset/component

In the Architecture Editor right-click on the core asset/component you want to configure. A context menu opens where you choose ’Configure...’. This opens the Asset Configu-

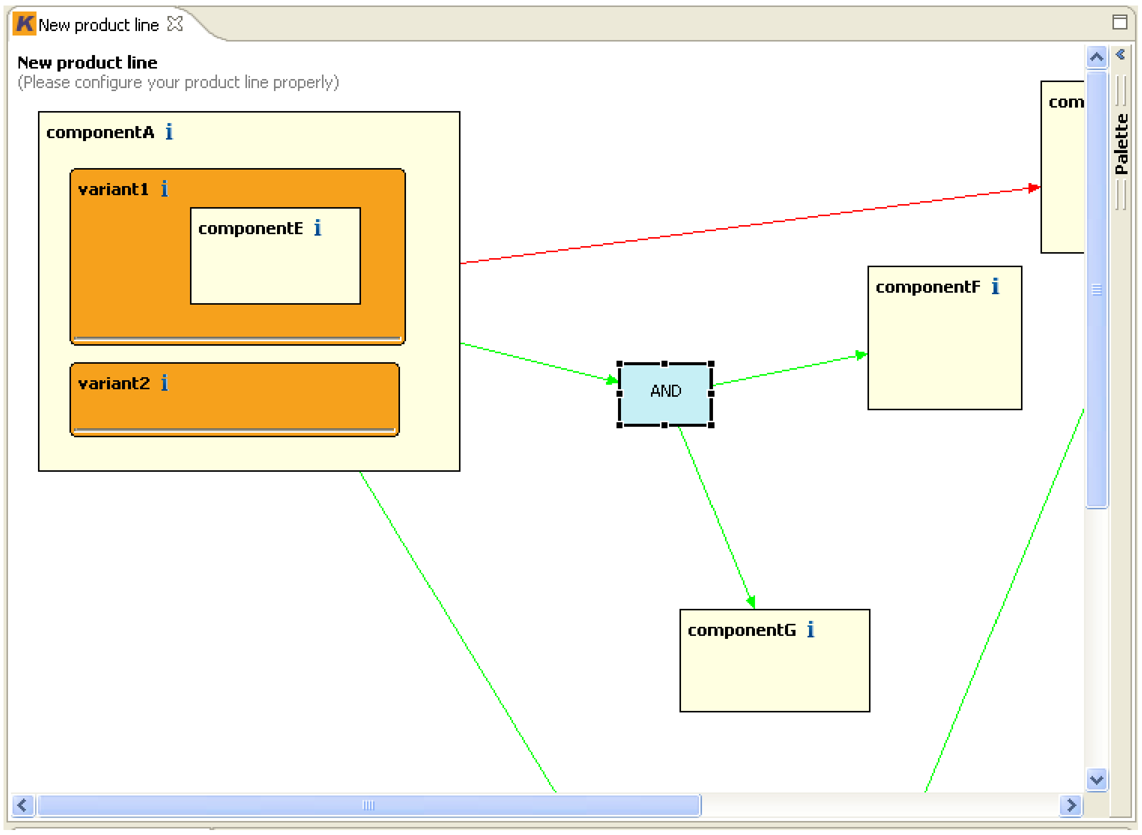


Figure 4.7: Architecture Editor

ration Dialog for components (see 4.10). Here you can change the name, description, maintainers and scripts for the asset. You can also set the asset on deprecated if you like.

Creating a variant

In the pallete select the ”variant” item. Click within a component in the Architecture Editor. A variant is inserted (see 4.11) and a dialog opens where you can enter the meta data of the variant. By clicking on the border and pulling it you can simply change the size of the variant.

Note: Variants can only be inserted into components.

Configuring a variant

In the Architecture Editor right-click on the variant you want to configure. A context menu opens where you choose ’Configure...’. This opens the Asset Configuration Dialog for

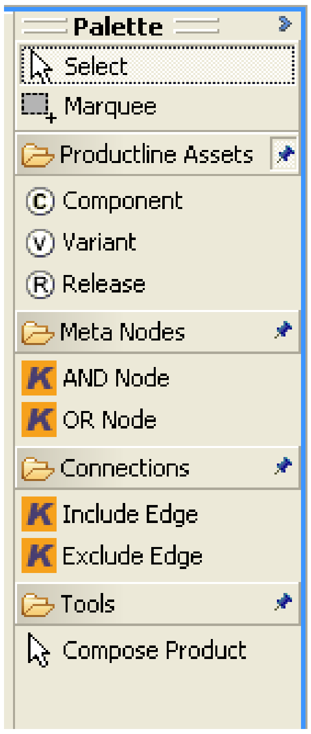


Figure 4.8: The palette

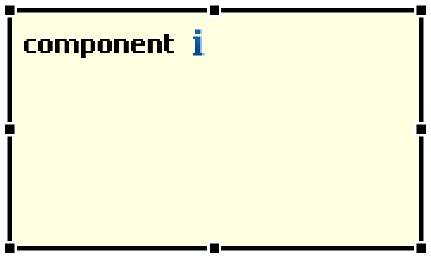
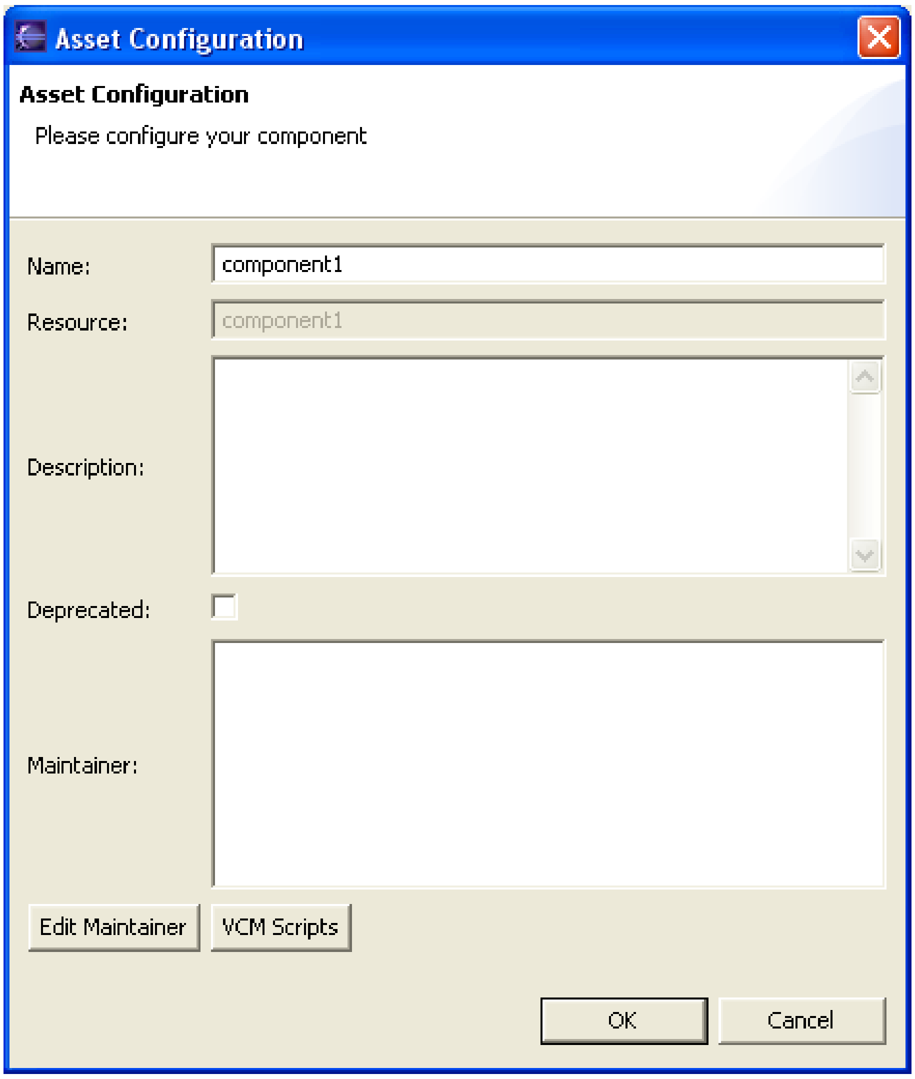


Figure 4.9: Component

variants (see 4.12). Here you can change the name, description and scripts for the asset. You can also set the asset on deprecated if you like.

Creating a release

In the pallete select the ”release” item. Click within a variant in the Architecture Editor. A release is inserted (see 4.13) and a dialog opens where you can enter the data of the release. For each file of the surrounding variant you can select the revision number you want to add to the release. By clicking on the border and pulling it you can simply change the size of the release.



Name: component1
Resource: component1
Description:
Deprecated:
Maintainer:

Figure 4.10: Asset Configuration Dialog for Components

Note: Releases can only be inserted into variants.

Configuring a release

In the Architecture Editor right-click on the release you want to configure. A context menu opens where you choose 'Configure...'. This opens the Asset Configuration Dialog for releases (see 4.14). Here you can change the name, description, resources and scripts for the asset. You can also set the asset on deprecated or mark it as a release if you like.

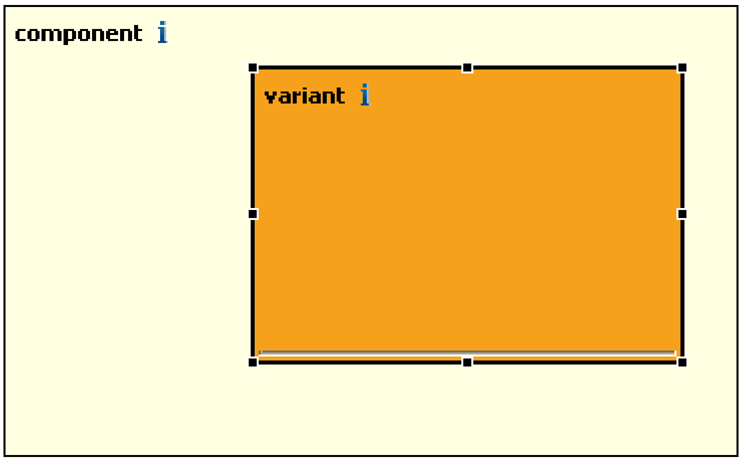
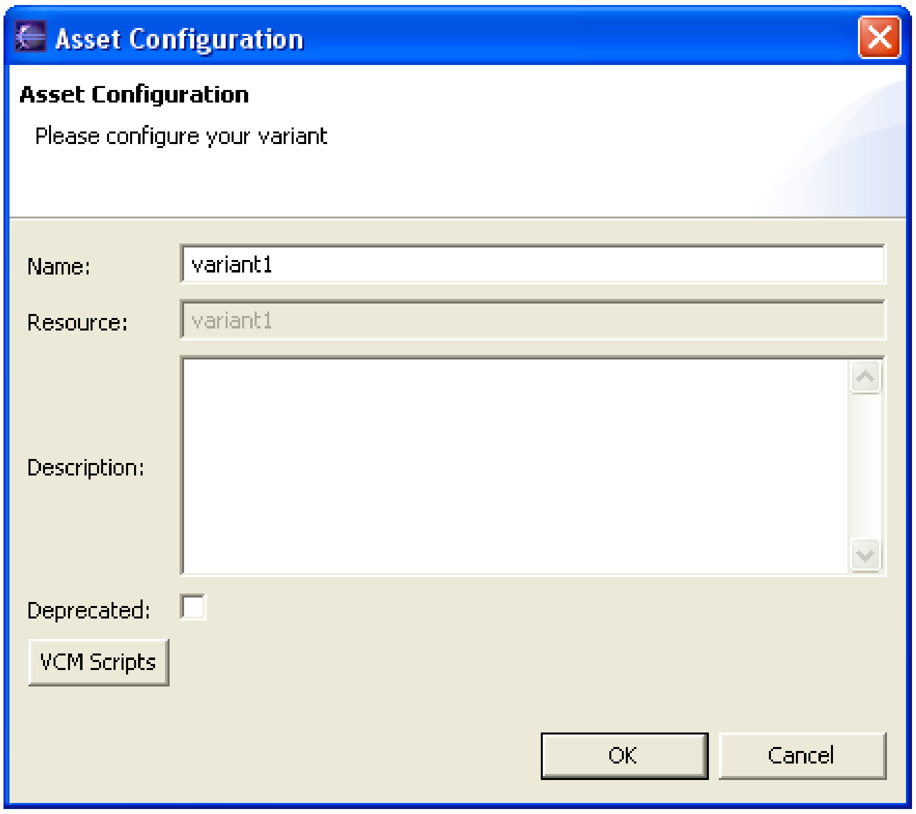


Figure 4.11: Variant



Asset Configuration

Please configure your variant

Name: variant1

Resource: variant1

Description:

Deprecated: ☐

Figure 4.12: Asset Configuration Dialog for Variants

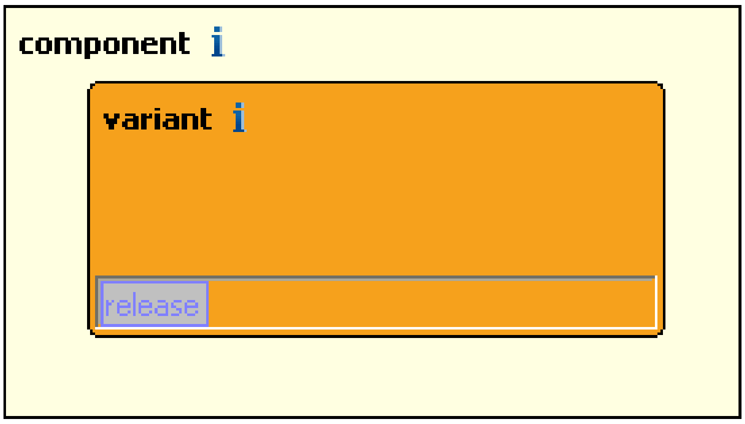


Figure 4.13: Release

Deleting a core asset/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.

Creating a meta node

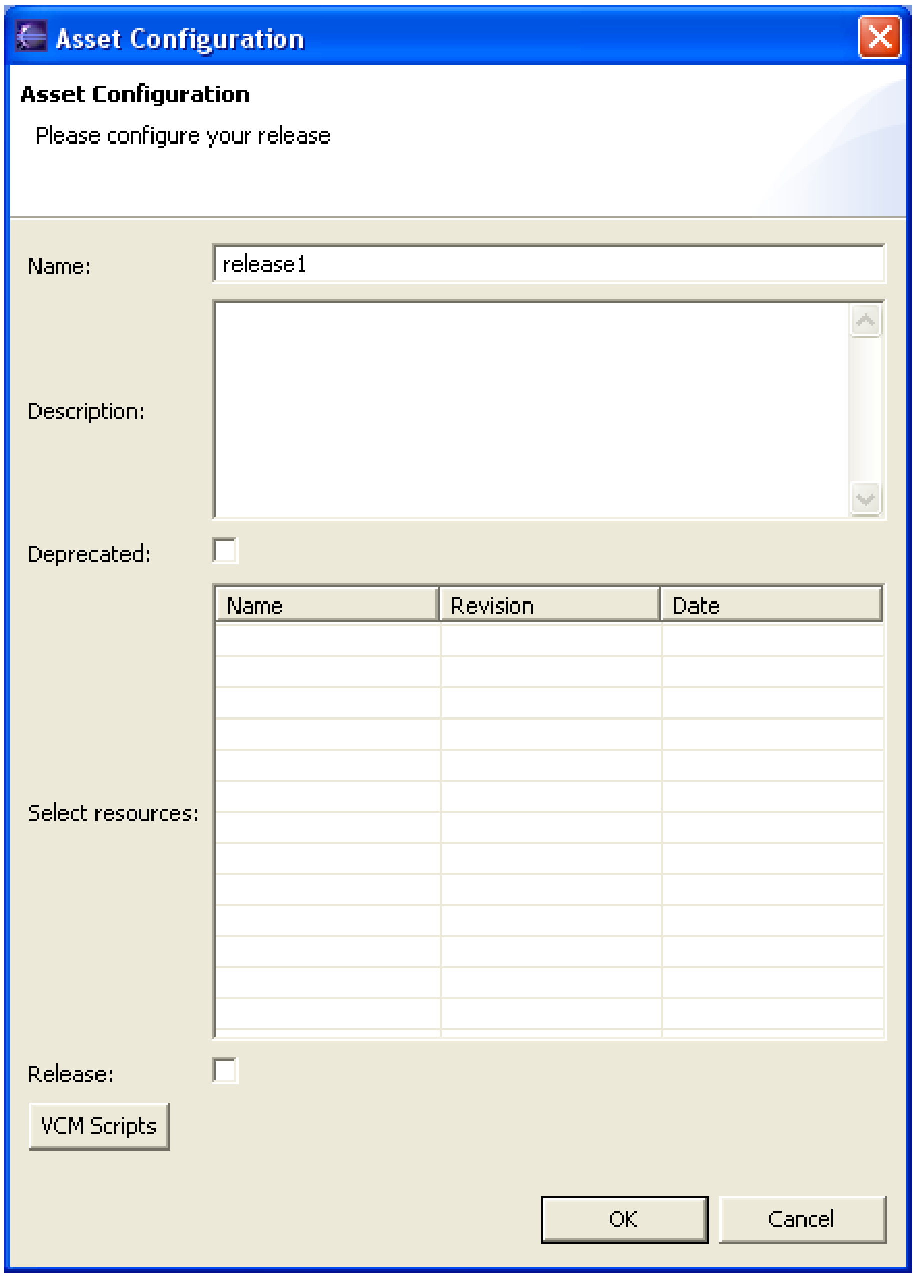
In the pallete select the ”meta node” item. Click within the Architecture Editor. A meta node is inserted (see 4.15). Possible meta node types are AND and OR. Be careful with the direction of the edges that are linked to a meta node! In the following architecture component A needs components B and C. But components B and C are independent of component A (see 4.16).

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.

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

Right-click on the object and choose ”configure”. The Asset Configuration dialog (see 4.23, 4.10, 4.12 and 4.14) 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 4.17).



Asset Configuration

Asset Configuration

Please configure your release

Name:	release1
Description:	
Deprecated:	☐
Select resources:	
Release:	☐

Name	Revision	Date

Figure 4.14: Asset Configuration Dialog for Releases

Creating a dependency edge

In the pallete select the ”include edge” item. Select an item in the Architecture Editor as the starting point. The next item you select will be the aiming point (see 4.18).

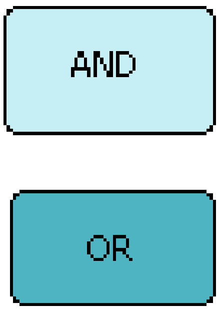


Figure 4.15: Some meta nodes

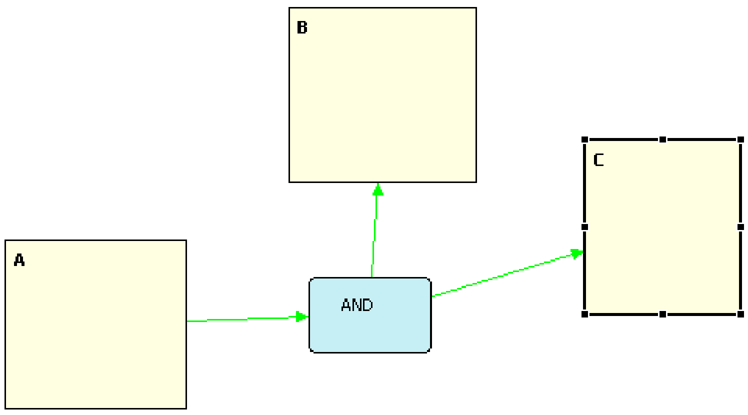


Figure 4.16: Meta node example

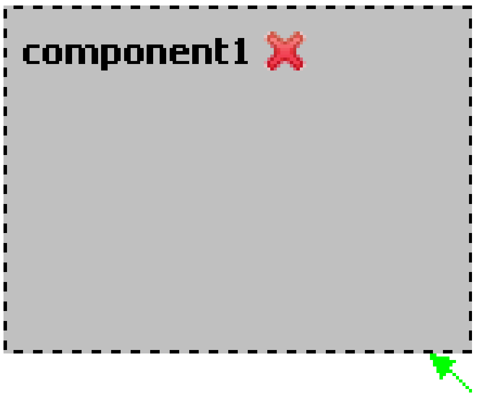


Figure 4.17: Deprecated component

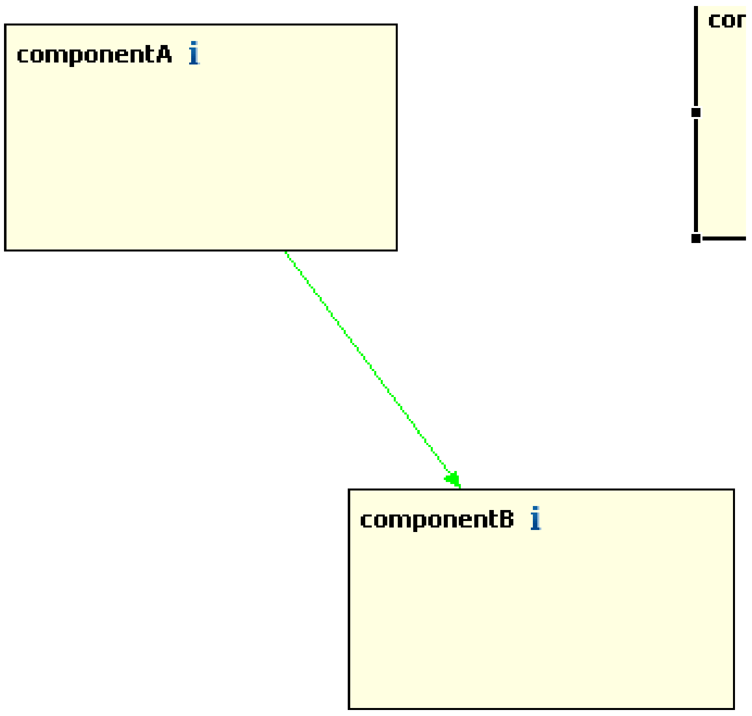


Figure 4.18: Dependency edge

Creating an exclusion edge

In the pallete select the ”exclude edge” item. Select an item in the Architecture Editor as the starting point. The next item you select will be the aiming point (see 4.19).

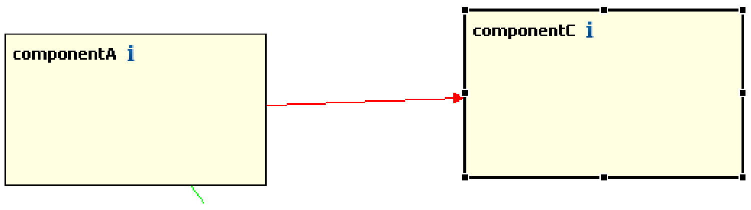


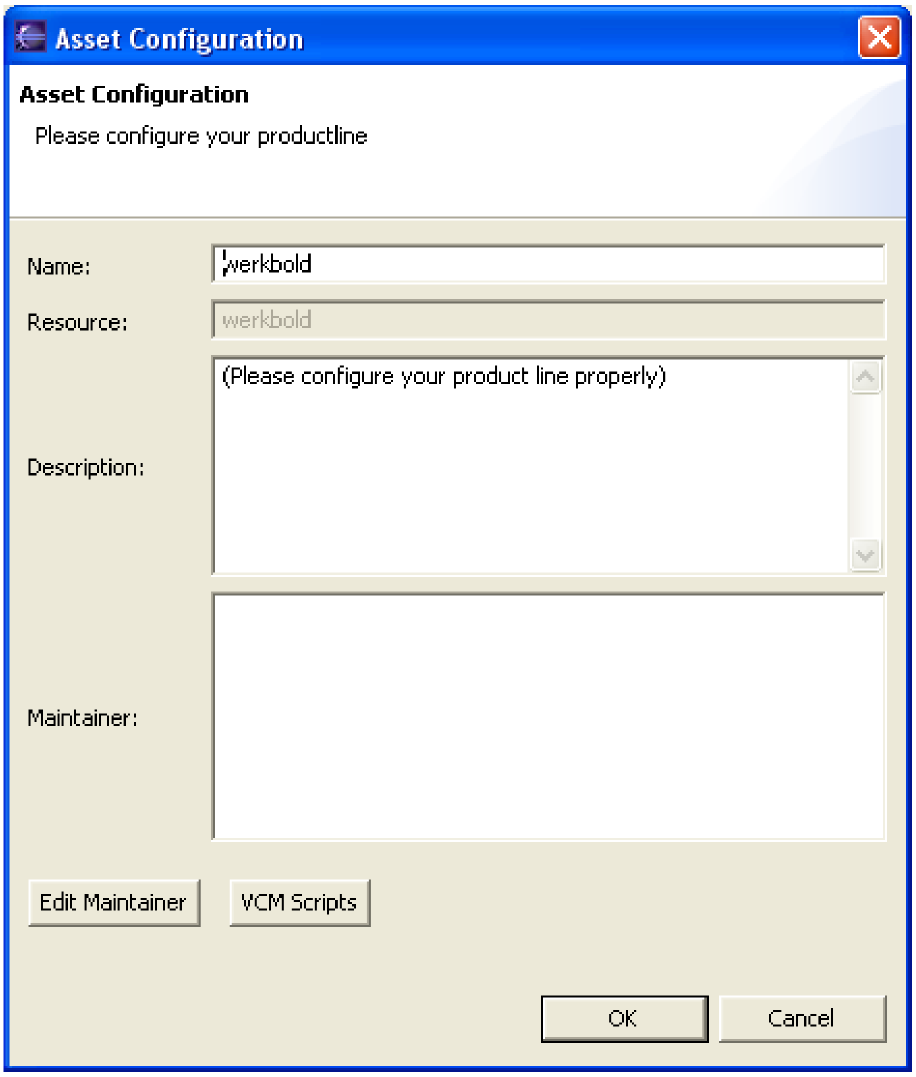
Figure 4.19: Exclusion edge

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.

Setting the maintainers of a productline, product or component

Right-click on the component or name of the product(line) and choose ”configure”. The Asset Configuration dialog (see 4.10, 4.23 and 4.20) opens. Press the ’Edit Maintainer’ button. A new window opens where you can select the users you want to make maintainers of the product(line)/component (see 4.21). After pressing ’OK’ you can see the new maintainers in the Asset Configuration dialog.


Asset Configuration

Asset Configuration

Please configure your productline

Name: werkbold

Resource: werkbold

Description: (Please configure your product line properly)

Maintainer:

Figure 4.20: Asset Configuration Dialog for Productlines

Moving an item

You can move any item by clicking on the item and pulling it.

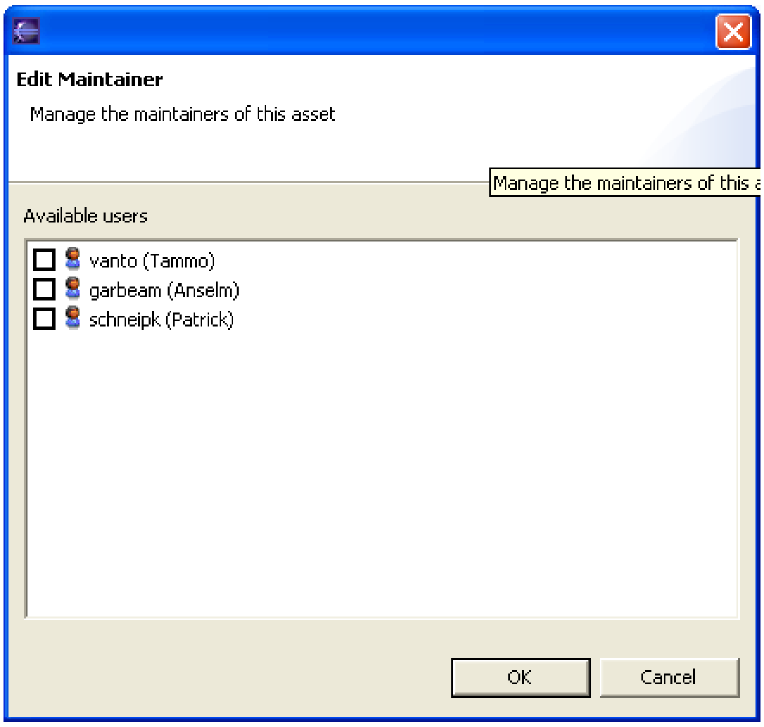


Figure 4.21: Selecting maintainers

Changing the size of an item

You can change the size of an item by clicking on the border of the item and pulling it.

Composing a product

In the palette press ’compose product’. The components, etc. in the architecture editor 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 4.22). When done press the ’create product’ button in the Architecture Editor. A dialog opens where you can enter the name, resource, description, maintainers, scripts and repository descriptor of your new product. After confirming the dialog, you can see your new product in the Architecture Tree and Architecture Editor.

Configuring a product

In the Architecture Editor right-click on the product you want to configure. A context menu opens where you choose ’Configure...’. This opens the Asset Configuration Dialog for products (see 4.23). Here you can change the name, resource, description,

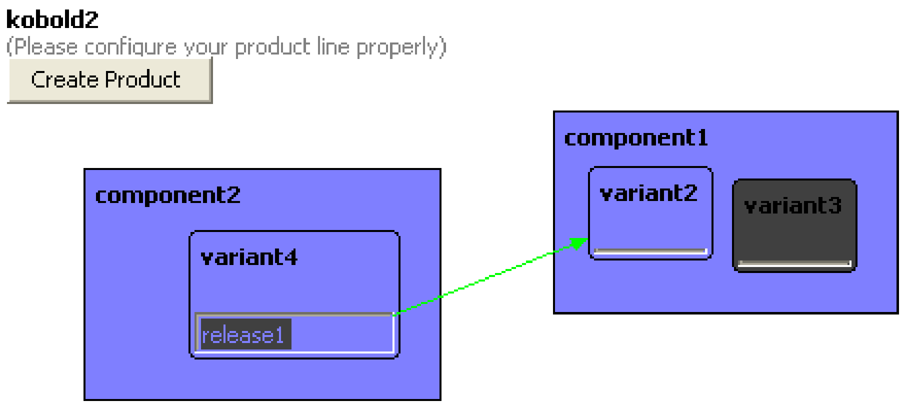


Figure 4.22: Composing a Product

maintainers, repository descriptor and scripts for the asset. You can also set the asset on deprecated if you like.

Editing the repository descriptor of a product

In the Architecture Editor right-click on the product. A context menu opens where you choose 'Configure Asset'. This opens the Asset Configuration dialog for products (see 4.23). Press the 'Repository...' button and a new dialog opens (see 4.24). Here you can change the data of the repository that saves the data of the product. Confirm with 'OK'.

Export

Select the productline, product, component or variant you want to export. In the context menu of the object select "export" (see 4.25). A wizard opens (see 4.26) 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.

4.1.3 Architecture Tree

The Architecture Tree shows the current projects that are checked out in your workspace (see 4.27). A project is always a productline. You can navigate between the different projects by selecting them. If you open the tree, you see the name of the productline. 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.

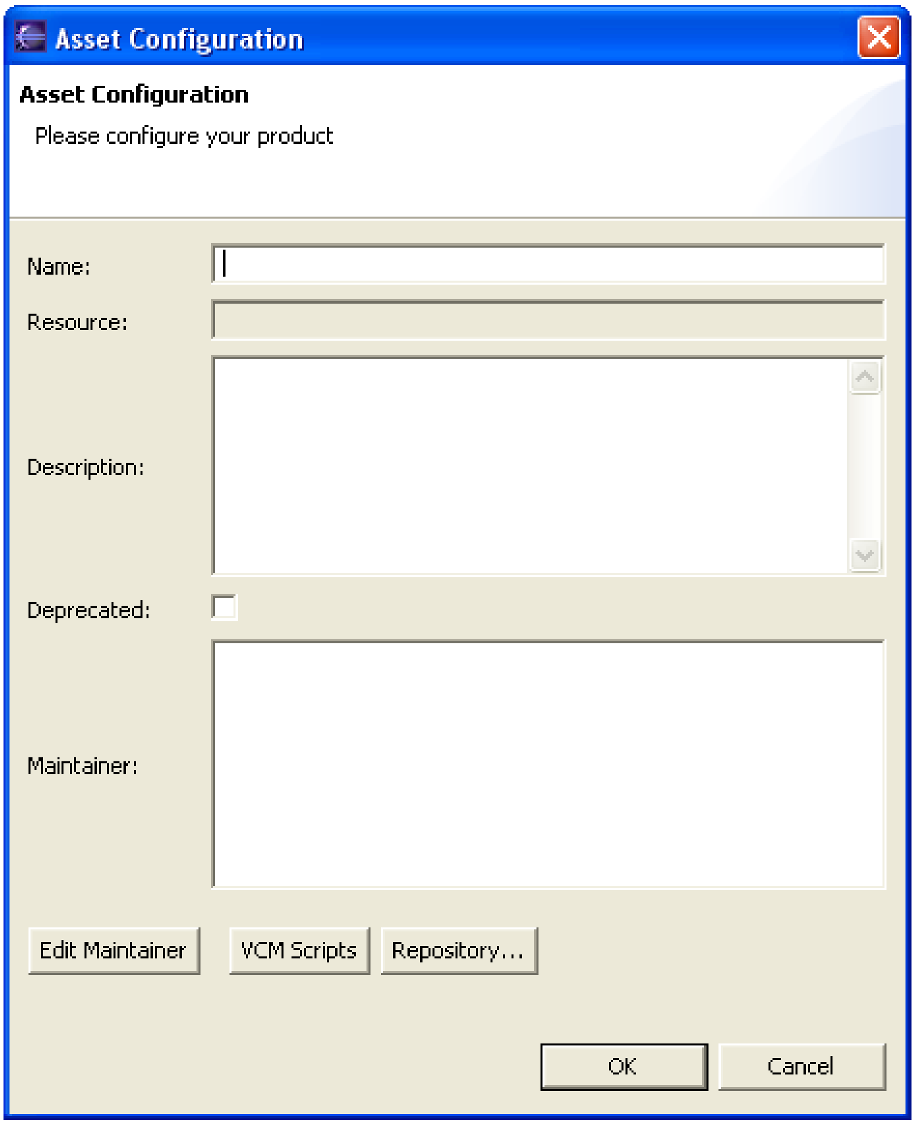
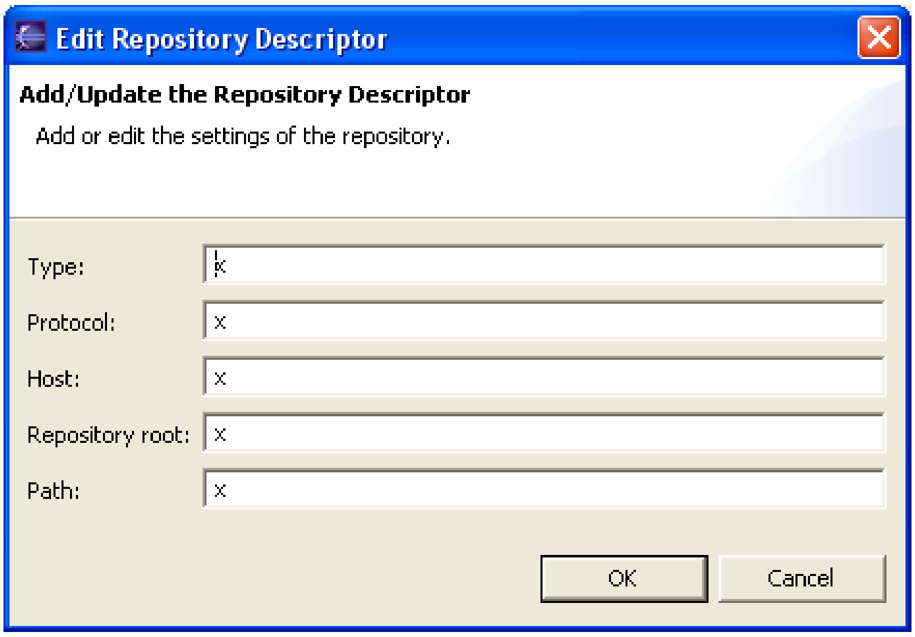


Figure 4.23: Asset Configuration Dialog for Products

When you right-click on a project, a context menu will open where you can choose between different actions (see 4.28 and 4.29). The possible actions are different depending on which asset you open the context menu on. Possible actions are:

- Create a new user
- Change password
- Update full name
- Configure asset
- Suggest an asset for a Core Group



Add/Update the Repository Descriptor

Add or edit the settings of the repository.

Type:	x
Protocol:	x
Host:	x
Repository root:	x
Path:	x

Figure 4.24: Edit Repository Descriptor

- Generate AssetInfo document
- Export
- Import
- Add to Product
- Delete Asset
- Version Control

Creating a user

Right-click in the architecture tree. In the context menu choose ’create new user’. The User Manager (see 4.30) opens where you see a list of all existing users. In order to create a new user, enter the username, name and password of the new user and press the ’add user’ button. You can see the new user in the list of the User Manager.

Changing your password

Right-click in the architecture tree and select ’update password’. A dialog opens where you can enter your new password (see 4.31). Confirm the password and press ’OK’. Your password has been changed.

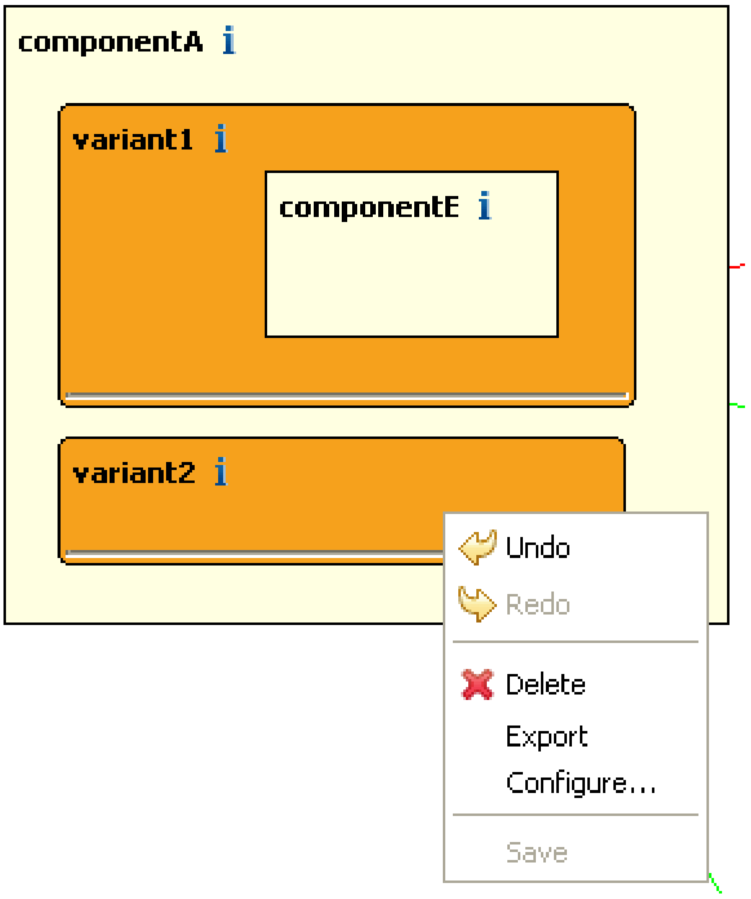
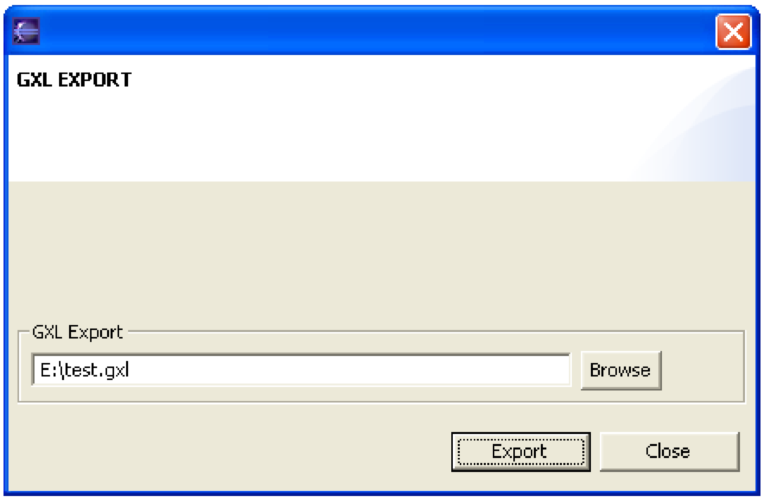


Figure 4.25: Context menu of the Architecture View



E:\test.gxl

Figure 4.26: Export wizard

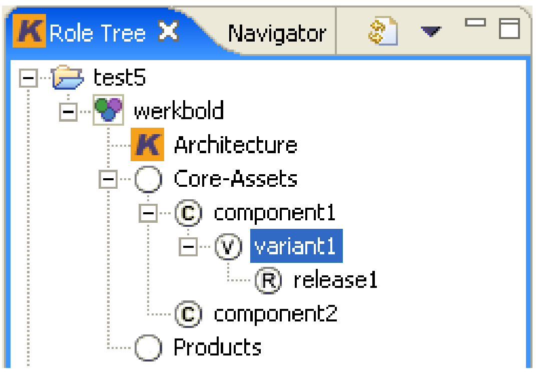


Figure 4.27: Architecture Tree

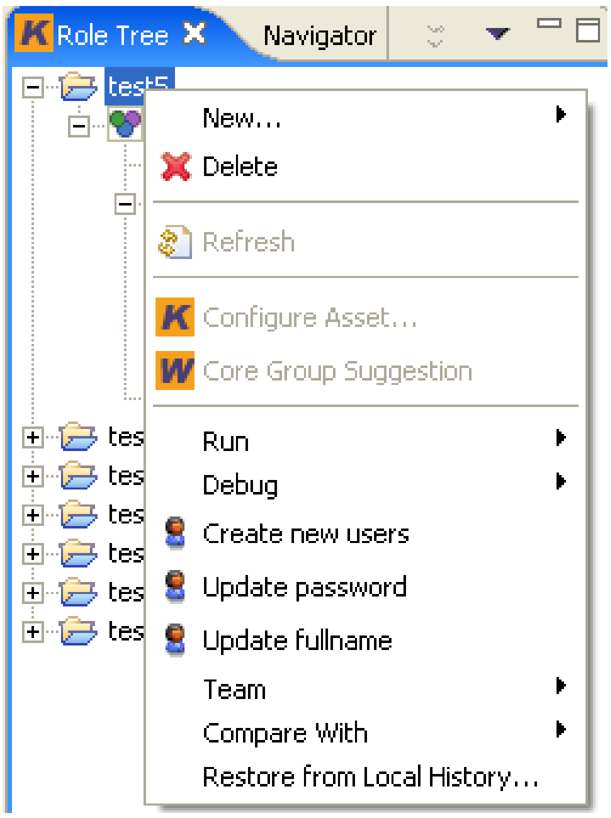


Figure 4.28: Architecture Tree Context Menu 1

Updating your full name

Right-click in the architecture tree and select ’update full name’. A dialog opens where you can enter your new full name (see 4.32). Confirm the changes with your password and press ’OK’. Your full name has been changed.

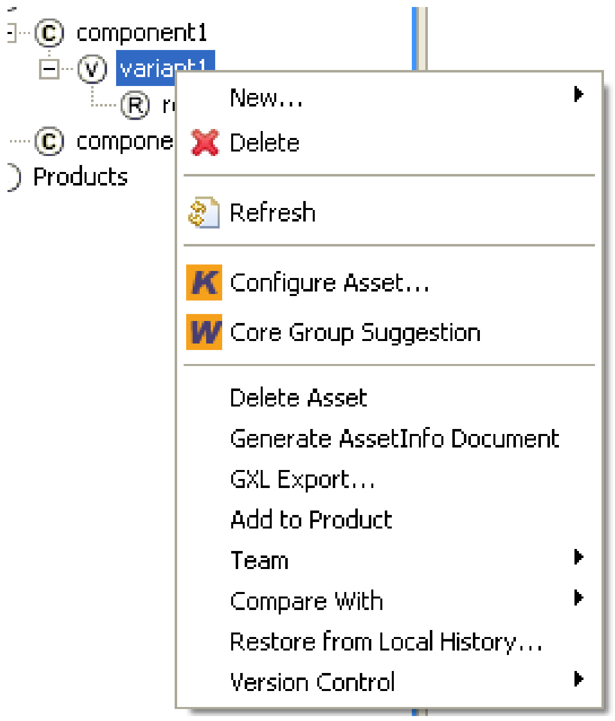


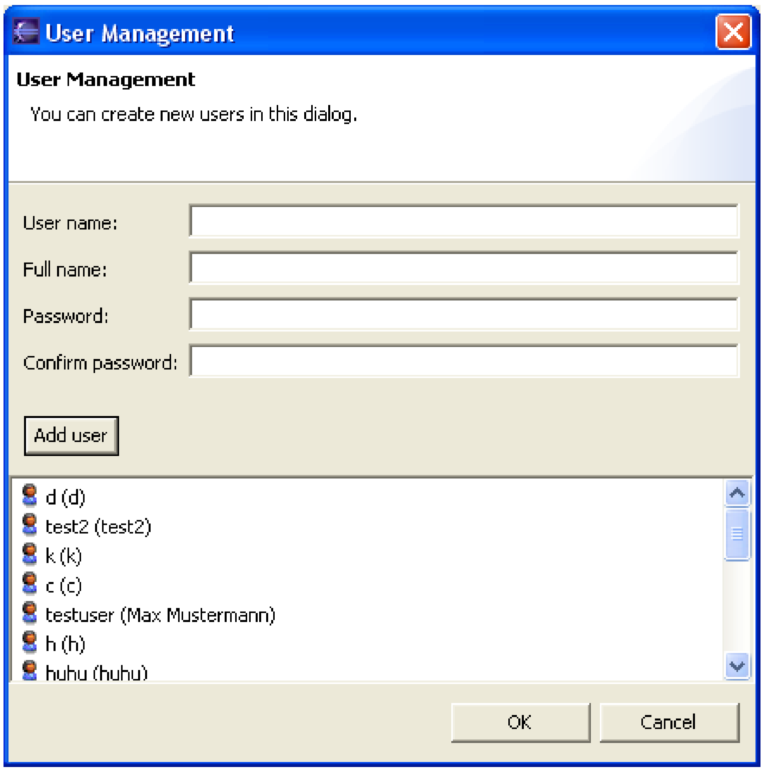
Figure 4.29: Architecture Tree Context Menu 2

Configure Asset

In the architecture tree right click on the asset you want to configure and select ’configure asset...’. The corresponding dialog will open. For more information see ’Architecture Editor’.

Suggesting an asset for a Core Group

In the Architecture Tree, 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, that is 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 ”Transmit” button. Pressing the ”Cancel” button will close the window without sending your message (see 4.44).


User Management

User Management

You can create new users in this dialog.

User name:

Full name:

Password:

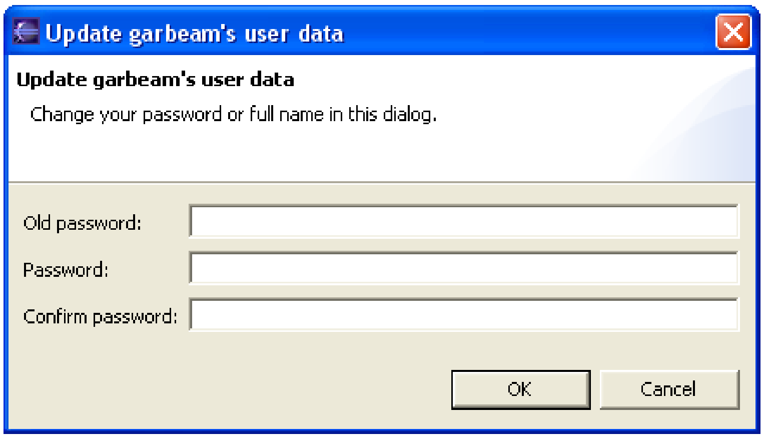
Confirm password:

Add user

d (d)
test2 (test2)
k (k)
c (c)
testuser (Max Mustermann)
h (h)
huhu (huhu)

OK Cancel

Figure 4.30: Create a new user


Update garbeam's user data

Update garbeam's user data

Change your password or full name in this dialog.

Old password:

Password:

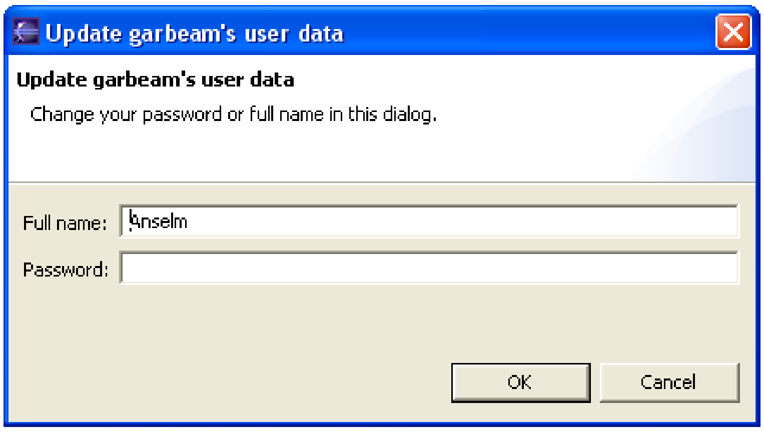
Confirm password:

OK Cancel

Figure 4.31: Change your password

Generating an AssetInfo document

In the architecture tree right-click on the object (productline, product, component, variant, release) you want the document about and select ’generate AssetInfo 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.


Update garbeam's user data

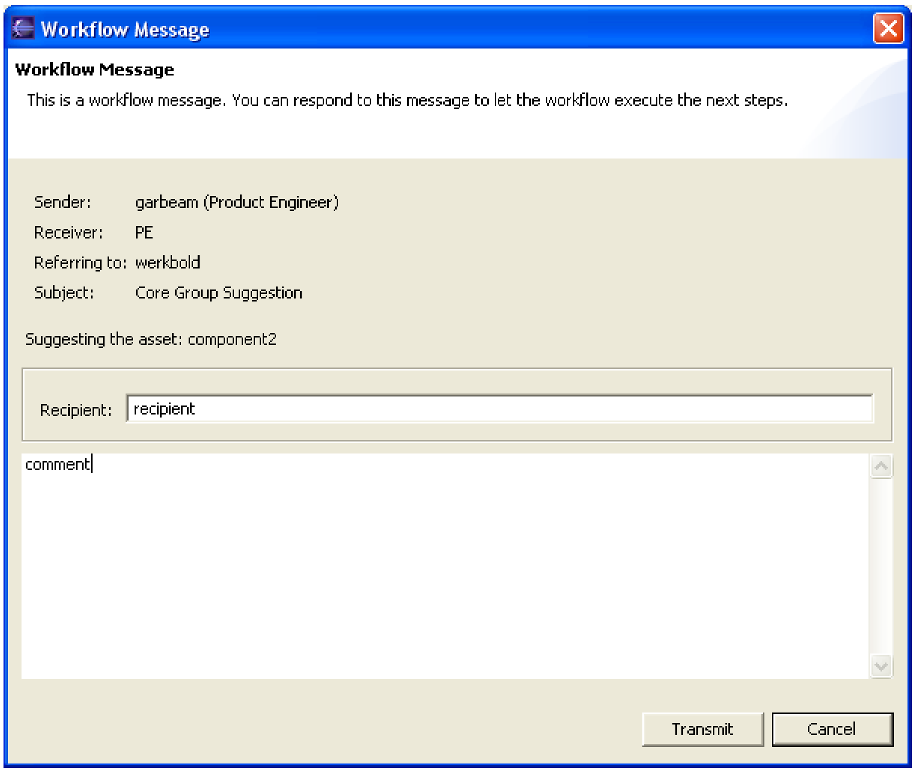
Update garbeam's user data

Change your password or full name in this dialog.

Full name: Anselm

Password:

Figure 4.32: Update your full name


Workflow Message

Workflow Message

This is a workflow message. You can respond to this message to let the workflow execute the next steps.

Sender: garbeam (Product Engineer)

Receiver: PE

Referring to: werkbold

Subject: Core Group Suggestion

Suggesting the asset: component2

Recipient: recipient

comment

Figure 4.33: Suggesting an asset for a Core Group

Export

In the architecture tree right-click on the asset you want to export and select ’GXL Export’. The corresponding dialog will open. For more information see ’Architecture Editor’.

Import

In the architecture tree right-click on a product and choose 'GXL Import...'. A dialog opens (see 4.34) where you can enter the name and path of the gxl-file you want to import. Click 'proceed' to finish.

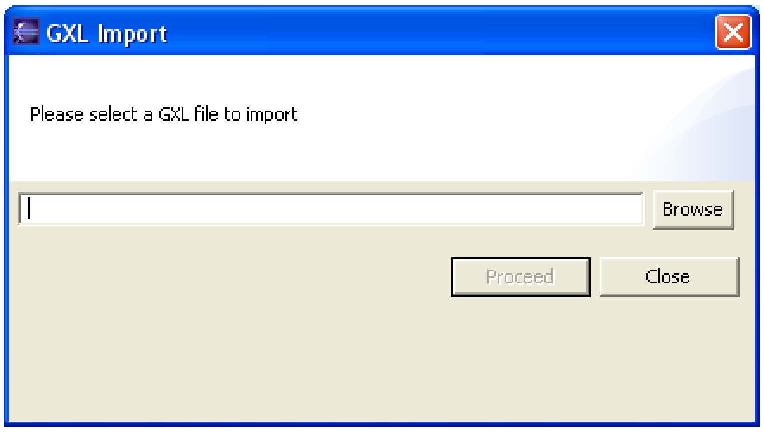


Figure 4.34: Import

Add Asset to Product

In the architecture tree right-click on the variant you want to add to a product and choose 'Add to Product'. A dialog opens where you can choose the products you want to add the variant to (see 4.35). When done confirm by pressing 'OK'. The variant is added to the chosen products.

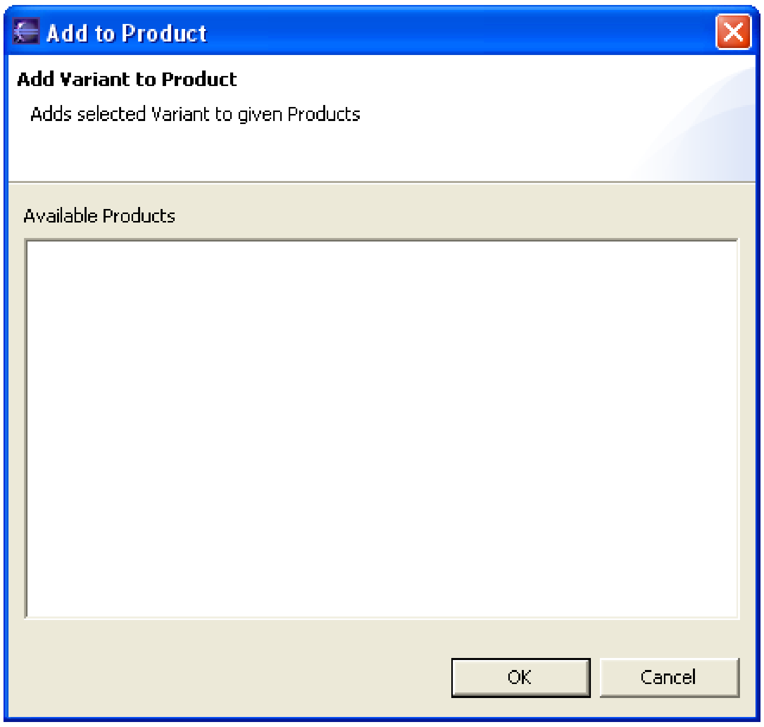
Delete Asset

In the architecture tree right-click on the asset you want to delete and choose 'delete asset'. A dialog opens (see 4.36) which asks you if you really want to delete the asset. If you press 'yes', the asset will be deleted. If you press 'no', the asset and its subassets will be marked as deprecated. You can change this status anytime through the asset configuration dialog.

Version Control

Right-click on an asset and choose 'Version Control'. Another menu opens where you can choose between different version control actions (see 4.37). Among those are:

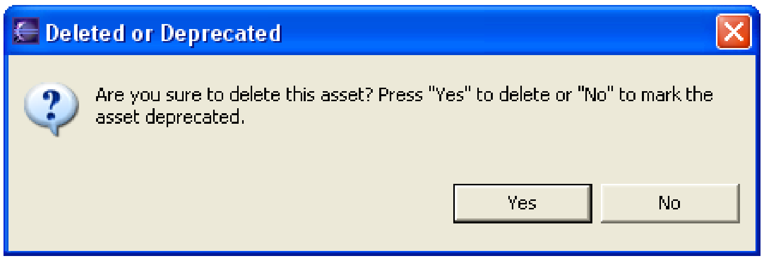
- Checkout



Add Variant to Product

Adds selected Variant to given Products

Figure 4.35: Add an Asset to a Product



Are you sure to delete this asset? Press "Yes" to delete or "No" to mark the asset deprecated.

Figure 4.36: Deleting an Asset

- Import
- Add
- Update
- Commit

4.1.4 Navigator

The Navigator hides behind the Architecture Tree and is brought to the front by clicking on it (see 4.38).

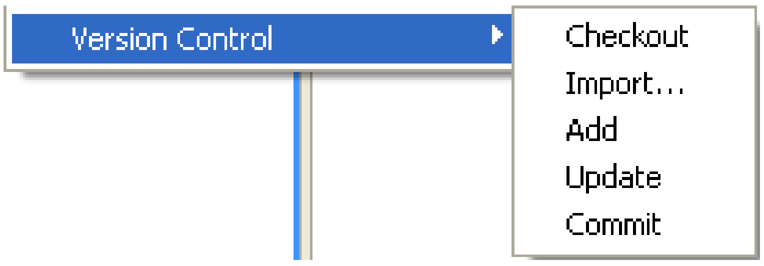


Figure 4.37: Version Control Actions

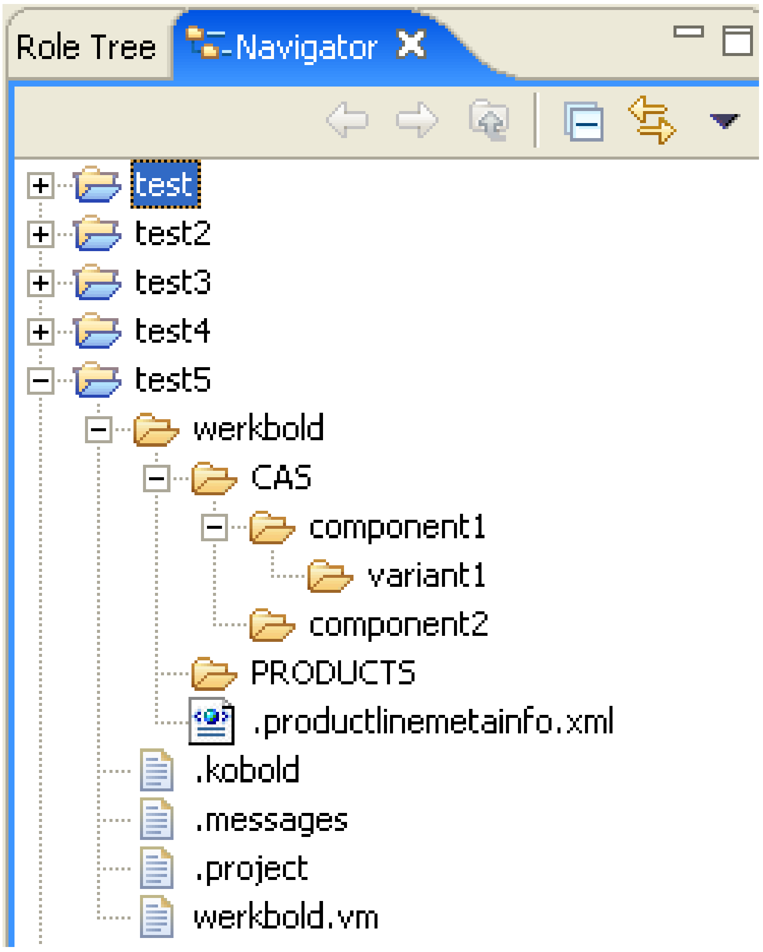
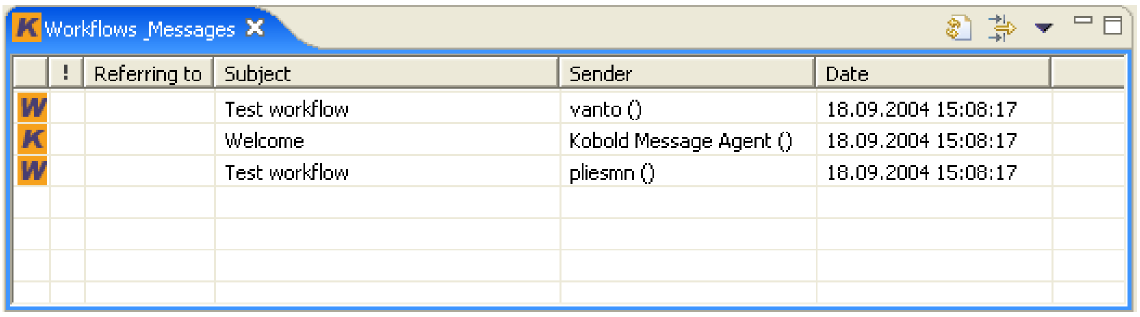


Figure 4.38: Navigator

It shows the folders the way they are saved on your hard disk and in the repositories. Here you can use the normal Eclipse actions you can easily look up in the Eclipse help.

4.1.5 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 4.39).

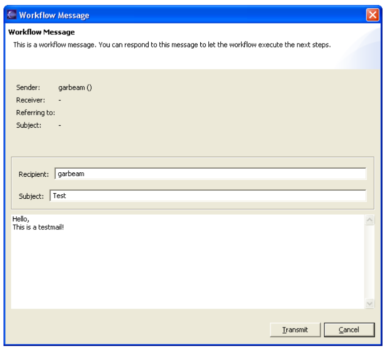


	!	Referring to	Subject	Sender	Date
W			Test workflow	vanto ()	18.09.2004 15:08:17
K			Welcome	Kobold Message Agent ()	18.09.2004 15:08:17
W			Test workflow	pliesmn ()	18.09.2004 15:08:17

Figure 4.39: Workflow/Task View

The icon in front of each message/workflow helps you to determine whether the entry is a workflow or a message. A 'W' represents a workflow, a 'K' represents a Kobold message.

Double-click on an entry and a separate dialog will be opened where you can read the details of that message or workflow (see 4.40).


Workflow Message

Workflow Message

This is a workflow message. You can respond to this message to let the workflow execute the next steps.

Sender: garbeam ()
Receiver: -
Referring to:
Subject: -

Recipient: garbeam
Subject: Test

Hello,
This is a testmail!

Figure 4.40: Message Dialog

When you right-click on a message, a context menu will open where you can choose between different actions (see 4.41). Among these are:

- Fetching messages
- Deleting the selected message

The Workflow View offers you the following additional options:

- Writing a mail
- Answering a mail

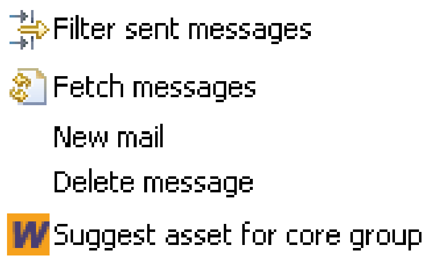


Figure 4.41: Message Context Menu

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

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.

Deleting a message

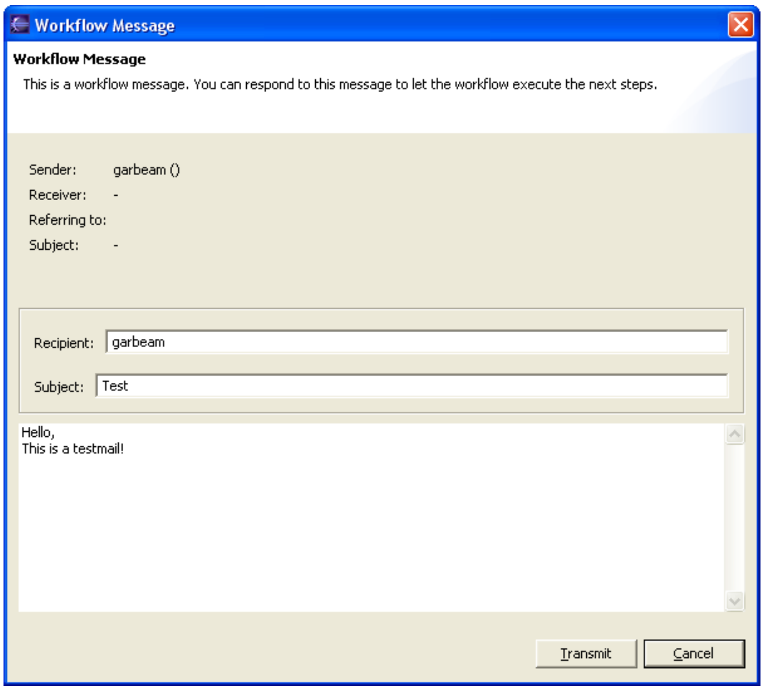
Right-click on the message and choose ”Delete message” in the context menu.

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 ”Transmit” button. Pressing the ”Cancel” button will close the window without sending your message (see 4.42).

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 ”Transmit” button. Pressing the ”Cancel” button will close the window without sending your message (see 4.43).



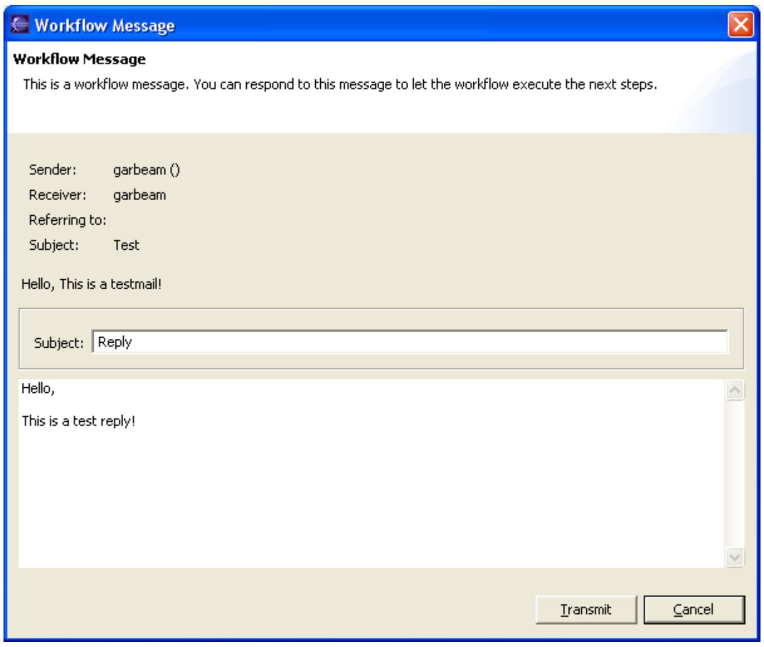
This is a workflow message. You can respond to this message to let the workflow execute the next steps.

Sender: garbeam ()
Receiver: -
Referring to:
Subject: -

Recipient: garbeam
Subject: Test

Hello,
This is a testmail!

Figure 4.42: Write a new mail



This is a workflow message. You can respond to this message to let the workflow execute the next steps.

Sender: garbeam ()
Receiver: garbeam
Referring to:
Subject: Test

Hello, This is a testmail!

Subject: Reply

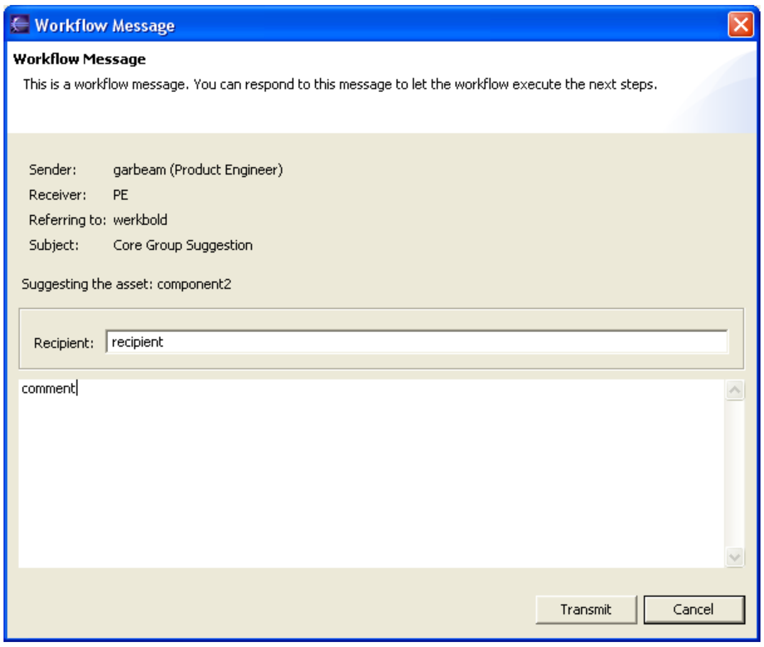
Hello,

This is a test reply!

Figure 4.43: Answer a mail

Suggesting 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, that is 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 "Transmit" button. Pressing the "Cancel" button will close the window without sending your message (see 4.44).



Workflow Message

This is a workflow message. You can respond to this message to let the workflow execute the next steps.

Sender:	garbeam (Product Engineer)
Receiver:	PE
Referring to:	werkbold
Subject:	Core Group Suggestion

Suggesting the asset: component2

Recipient: recipient

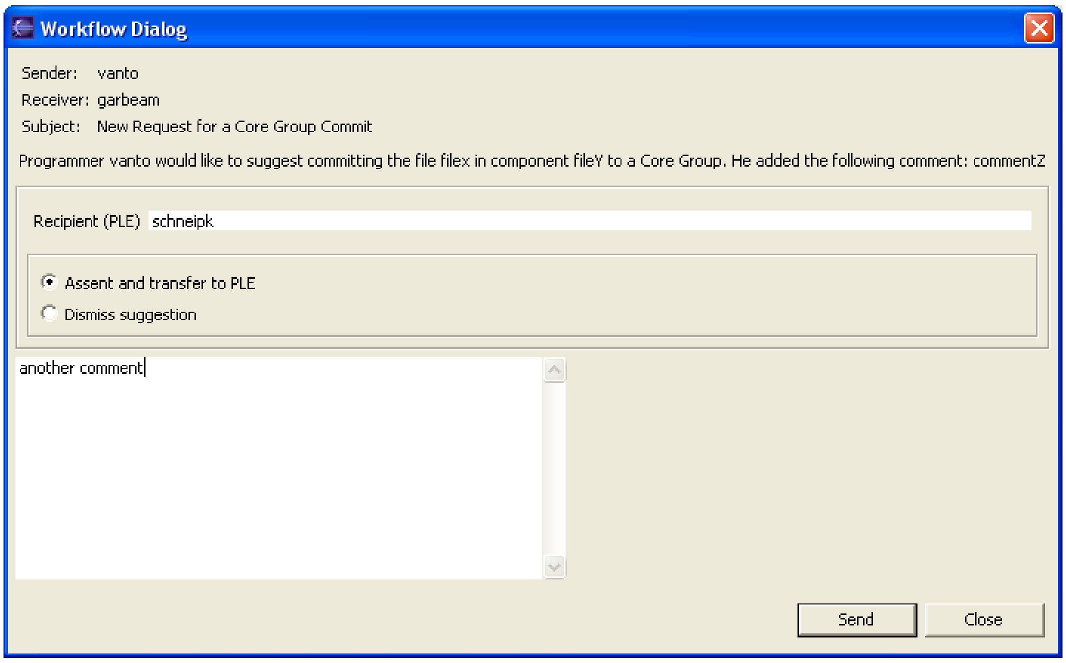
comment

Figure 4.44: Suggesting a file for a Core Group

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 4.45) or PE (see 4.46). 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 "Transmit" 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.



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

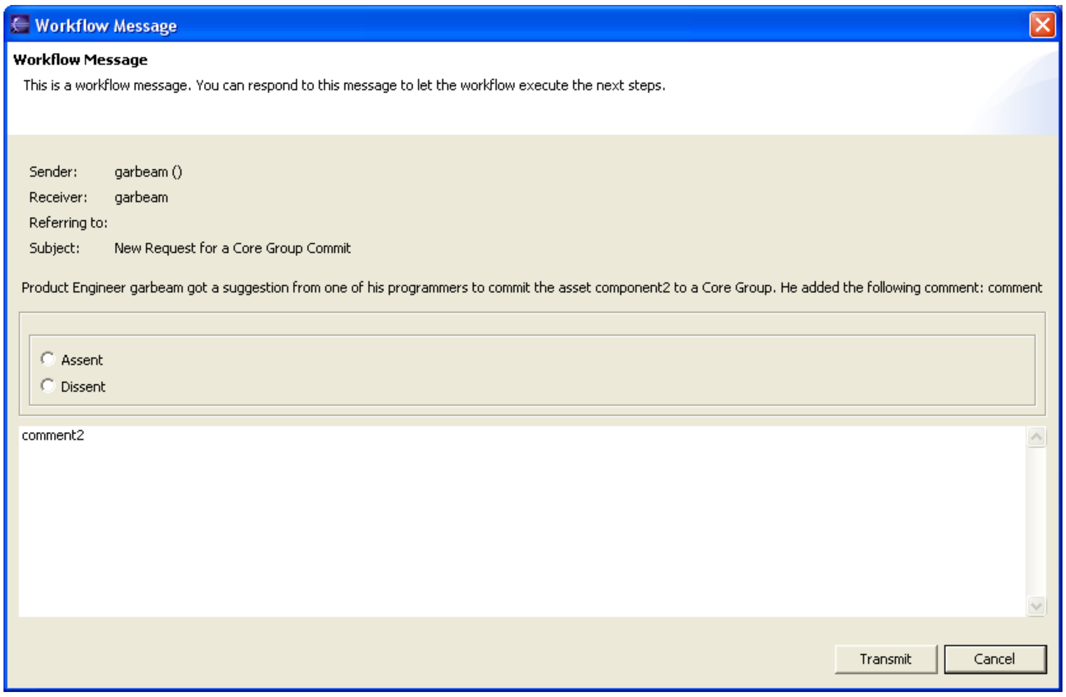
Programmer vanto would like to suggest committing the file filex in component fileY to a Core Group. He added the following comment: commentZ

Recipient (PLE) schneipk

(●) Assent and transfer to PLE
( ) Dismiss suggestion

another comment

Figure 4.45: Dealing with a Core Group suggestion - PE



Workflow Message
This is a workflow message. You can respond to this message to let the workflow execute the next steps.

Sender: garbeam ()
Receiver: garbeam
Referring to:
Subject: New Request for a Core Group Commit

Product Engineer garbeam got a suggestion from one of his programmers to commit the asset component2 to a Core Group. He added the following comment: comment

( ) Assent
( ) Dissent

comment2

Figure 4.46: Dealing with a Core Group suggestion - PLE

4.1.6 Minimap

The Minimap (see 4.47) is on the left side beneath the Architecture Tree. It shows the whole architecture. By clicking at one spot on the map, the Architecture Editor automatically centers on that spot. You can also click on the coloured rectangle and pull it to another position. By doing this you can easily navigate through your architecture.

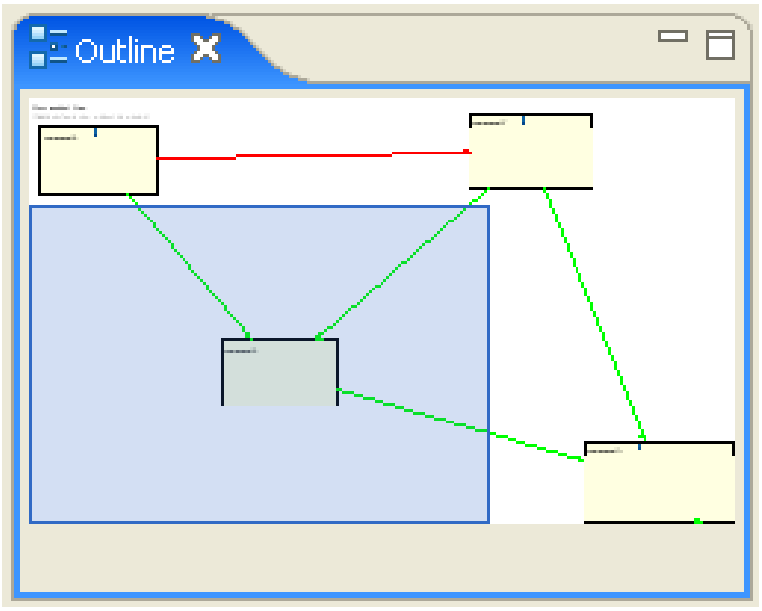


Figure 4.47: Minimap

4.2 Server

The Server administrates the data of each user and their responsibilities. Also it offers the Clients a central message service . If a message is sent from a Client to the Server, the Server checks the message for possible consequences which are assigned to other clients as workflows. Besides the Server administrates the paths and the access configurations of the repositories which save the data of the products and the productlines.

4.2.1 Changing the Server Properties

When installing the server properties are automatically set for you. However, this section is for those of you who want to change your server properties manually:

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=[your storage path]
kobold.server.productStore=product.xml
kobold.server.userStore=user.xml
kobold.server.ruleset={path to ruleset]
#
java.protocol.handler.pkgs=com.sun.net.ssl.internal.www.protocol
javax.net.debug=none#all
javax.net.ssl.keyStore=[your keystore path]
javax.net.ssl.keyStorePassword=[your keystore passphrase]
javax.net.ssl.trustStore=[your truststore path]
javax.net.ssl.trustStorePassword=[your truststore passphrase]
```

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.productStore	file name to store products and productlines
kobold.server.userStore	file name to store user data
kobold.server.ruleset	path to your ruleset file
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

4.2.2 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 examples.

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”>

    <parameter identifier = ”spy”>

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

    </parameter>
    <java:condition> spy.getMethodName().equals(”addUser”) </java:condition>
    <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 sample rules for all actions.

In the last chapter you find the source code of the classes you need to work with for writing your own rules.

4.3 Server Administration Tool (SAT)

Kobold servers do not have to be administrated directly. Instead the Kobold System provides a stand-alone client with the sole purpose of administrating Kobold servers over a secure ssl-connection.

After having set the server’s url and its administration password you have the following possibilities:

- create a productline
- delete a productline
- create a new user
- remove a user
- assign an existing user to a productline as PLE
- remove a user’s PLE rights for a productline
- get a list of all productlines
- get a list of all PLEs of a productline
- get a list of all users

4.3.1 Setting up the server connection

At startup the SAT tries per default to establish a connection to a server running at ’localhost’ on port 23232 without a set administration password. You can change those default settings at any time by entering the command ’setserver’.

After entering that command the current properties of the server connection are listed (note that the password is not shown in plain text). Enter ’1’ to change the server’s ip, ’2’ to change the server’s port and ’3’ to change the administration password.

If you decide to set or change a password the SAT provides secure password masking by prompting a region of 8 randomly changing characters into which your input is - randomly - echoed to. This happens because it is not possible to prevent the java input stream from echoing to the console. Nonetheless echoing into that region makes it virtually impossible to determine which of the prompted characters are echoed and which are generated randomly.

When you have finished modifiying the connection properties confirm by entereing ’p’. The SAT tries to establish the new connection. Enter ’c’ to cancel the action.

4.3.2 Creating a productline

Enter the command ’newpl’. A listing is shown with the properties of the lately created productline. You can change the properties as described above. By entering ’p’ the SAT tries to create a new productline with the current settings.

Please note that every productline of a Kobold server must have an unique name. Creation of a productline with an already assigned name will be refused by the server.

4.3.3 Deleting a product line

Enter the command ’rempl’. All registered productlines’ names are listed and you are asked to enter the name of the productline you want to remove. Once you have confirmed your entry, the productline is deleted.

4.3.4 Assigning an existing user to a productline as PLE

Enter the command ’assignple’. You are asked to enter the name of the person you want to become PLE. Finally, enter the name of the productline you want to assign the new PLE to. Once you confirmed your entry, the user has PLE rights.

4.3.5 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 productline you want to remove the PLE from. Once you confirmed your entry, the user is no longer PLE of the entered product line.

4.3.6 Getting a list of all existing commands

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

4.3.7 Getting information about SAT’s version and licence

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

4.3.8 Creating a new user

Enter the command ’newuser’. A listing is shown with the properties of the lately created user, his/her username, fullname and initial password. After enetering ’p’ a new user with the shown properties is created on the server.

Note that every registered user needs to have its own unique username. Creation of a new user with a username that’s already been registered will be refused by the server.

4.3.9 Deleting a user

Enter the command ’remuser’. A listing with all registered usernames is shown and you are asked to enter the username of the user you like to remove from the server. A check is performed to determine if the specified user is still assigned to an asset. If so, you are informed by the SAT and given the possibility to cancel the action.

4.3.10 Getting a list of all productlines

Enter the command ’pllist’ in order to get a list of all productlines registered on the current server.

4.3.11 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.

4.3.12 Getting a list of all registered users

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

4.3.13 Exiting the tool

Enter the command ”exit”.

4.3.14 Changing the SAT Properties

When installing the SAT properties are automatically set for you. However, this section is for those of you who want to change your SAT properties manually:

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=none #all
javax.net.ssl.keyStore=/home/stsopra/werkbold/contan/workspace/kobold.com
javax.net.ssl.keyStorePassword=kobold1
javax.net.ssl.trustStore=/home/stsopra/werkbold/contan/workspace/kobold.c
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**
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

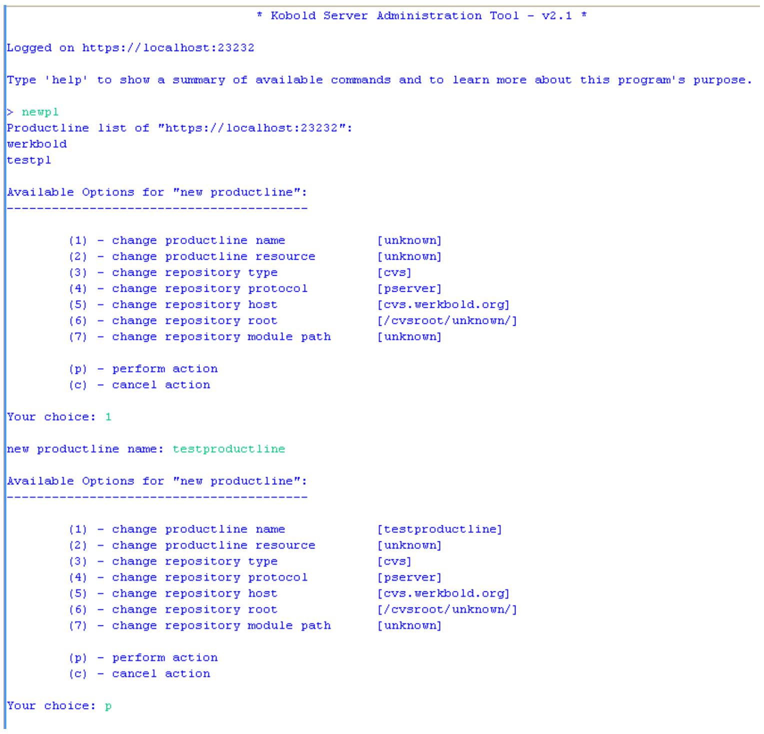
5 12-step Tutorial

5.1 Preconditions

In order to perform this tutorial, install Kobold and start the Server and Server Administration Tool. Make sure that you have set the correct properties for your Client.

5.2 Creating a productline

In your SAT Tool, enter the command ’newpl’. A listing is shown with the properties of the lately created productline. Enter the command ’1’ and then ’testproductline’ as name for your new productline. Enter the command ’p’ to create the productline (see 5.1).

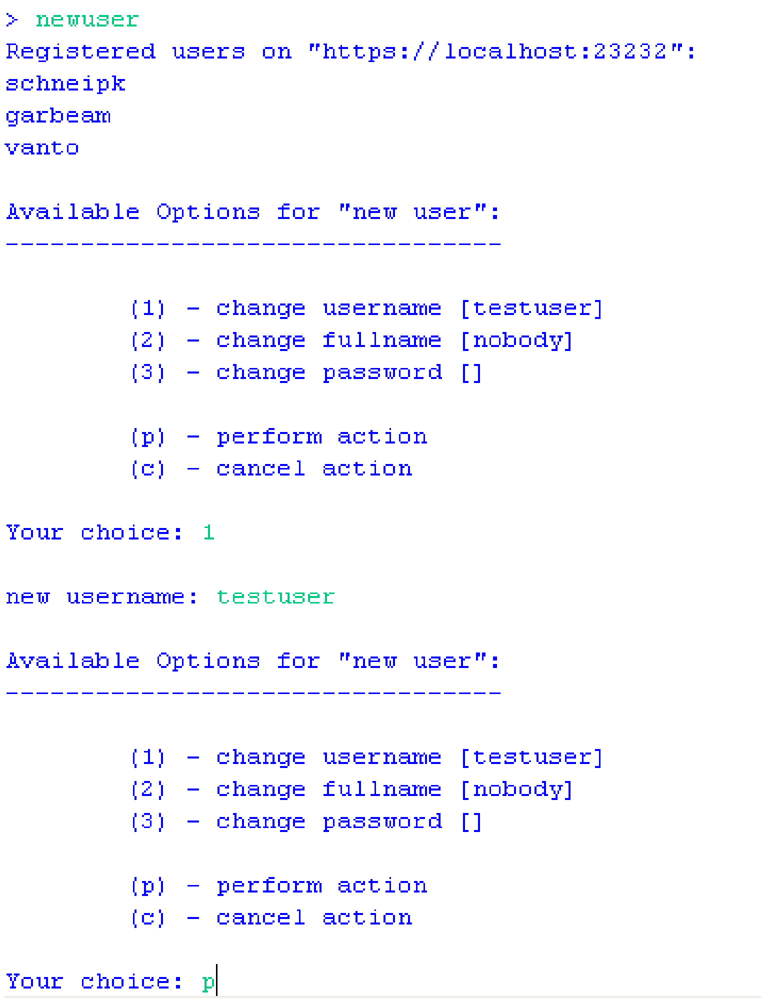


```
                                 * Kobold Server Administration Tool - v2.1 *

Logged on https://localhost:23232

Type 'help' to show a summary of available commands and to learn more about this program's purpose.

> newpl
Productline list of "https://localhost:23232":
werkbold
testpl

Available Options for "new productline":
----------------------------------------

        (1) - change productline name            [unknown]
        (2) - change productline resource        [unknown]
        (3) - change repository type             [cvs]
        (4) - change repository protocol         [pserver]
        (5) - change repository host             [cvs.werkbold.org]
        (6) - change repository root             [/cvsroot/unknown/]
        (7) - change repository module path      [unknown]

        (p) - perform action
        (c) - cancel action

Your choice: 1

new productline name: testproductline

Available Options for "new productline":
----------------------------------------

        (1) - change productline name            [testproductline]
        (2) - change productline resource        [unknown]
        (3) - change repository type             [cvs]
        (4) - change repository protocol         [pserver]
        (5) - change repository host             [cvs.werkbold.org]
        (6) - change repository root             [/cvsroot/unknown/]
        (7) - change repository module path      [unknown]

        (p) - perform action
        (c) - cancel action

Your choice: p
```

Figure 5.1: Creating a productline

5.3 Creating a new user

In your SAT Tool, enter the command ’newuser’. A listing is shown with the properties of the lately created user, his/her username, fullname and initial password. Enter the command ’1’ and then ’testuser’ as username for your new user. Enter the command ’p’ to create the user (see 5.2).

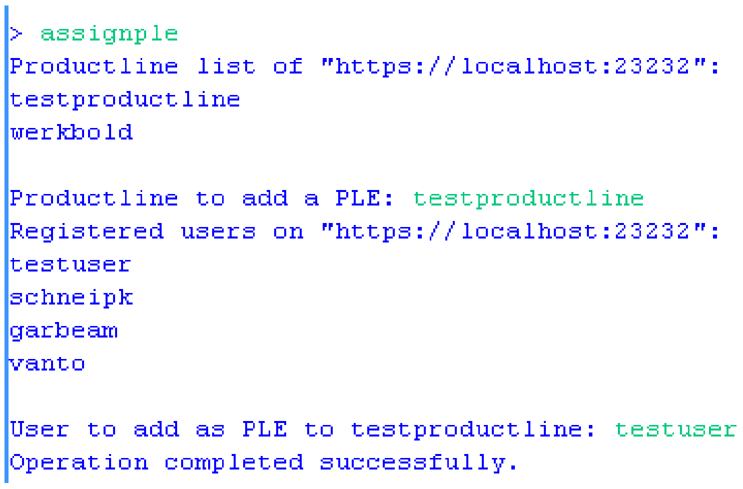


```
> newuser
Registered users on "https://localhost:23232":
schneipk
garbeam
vanto

Available Options for "new user":
---------------------------------

        (1) - change username [testuser]
        (2) - change fullname [nobody]
        (3) - change password []

        (p) - perform action
        (c) - cancel action

Your choice: 1

new username: testuser

Available Options for "new user":
---------------------------------

        (1) - change username [testuser]
        (2) - change fullname [nobody]
        (3) - change password []

        (p) - perform action
        (c) - cancel action

Your choice: p
```

Figure 5.2: Creating a new user

5.4 Assigning an existing user to a productline as PLE

In your SAT Tool, enter the command ’assignple’. As PLE enter the username ’testuser’. As productline enter your productline ’testproductline’ (see 5.3).



```
> assignple
Productline list of "https://localhost:23232":
testproductline
werkbold

Productline to add a PLE: testproductline
Registered users on "https://localhost:23232":
testuser
schneipk
garbeam
vanto

User to add as PLE to testproductline: testuser
Operation completed successfully.
```

Figure 5.3: Assigning your ple to your productline

5.5 Starting a new project

Start the Client. In the File menu select ’New’ and then ’Kobold PLAM Project’. The Kobold wizard opens. Enter the url of your Kobold server, your username (testuser) and a blank for your password. Then press ”test connection”. If the test succeeds, the ”next” button is enabled (see 5.4).

After that you choose ’testproductline’ as the productline you want to check out (see 5.5).

In the last step you have to enter ’testproject’ as the name of the project you want to create (see 5.6).

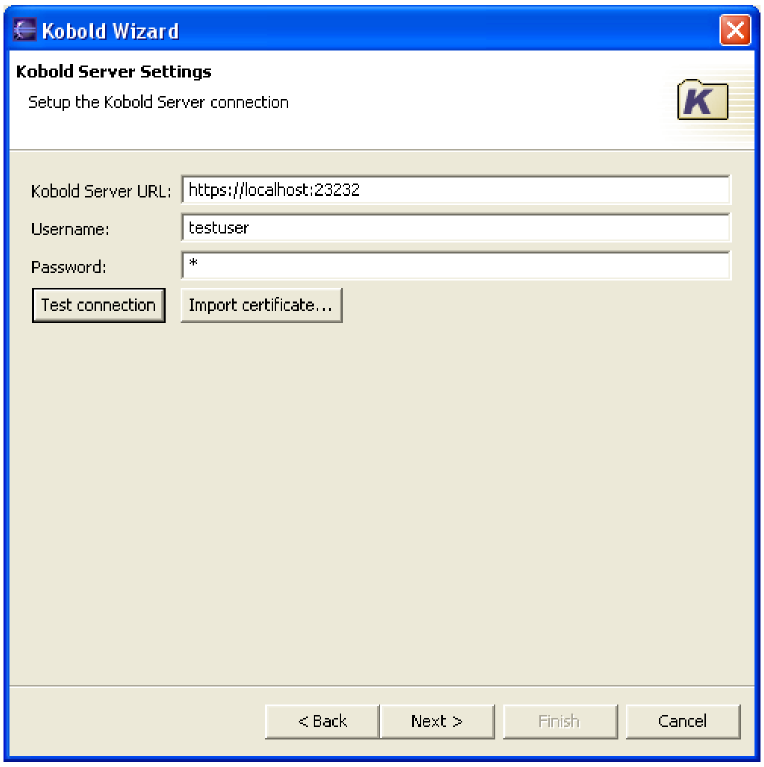
Press finish and your new project is created.

5.6 Creating a Core Assets

In the Architecture Tree (on the left) open up the tree of your new project. Double-click on ’architecture’ in order to open the Architecture Editor.

Open the pallete on the right of the Architecture Editor and select the ’component’ item. Click in the Architecture Editor. A core asset is inserted and a dialog opens where you enter ’componentA’ as name of the core asset (see 5.7).

Repeat these steps another two times and create the core assets ’componentB’ and ’componentC’. After this your architecture should look like the following (see 5.8):


Kobold Wizard

Kobold Server Settings

Setup the Kobold Server connection

Kobold Server URL: https://localhost:23232

Username: testuser

Password: *

Figure 5.4: Kobold wizard

5.7 Creating variants

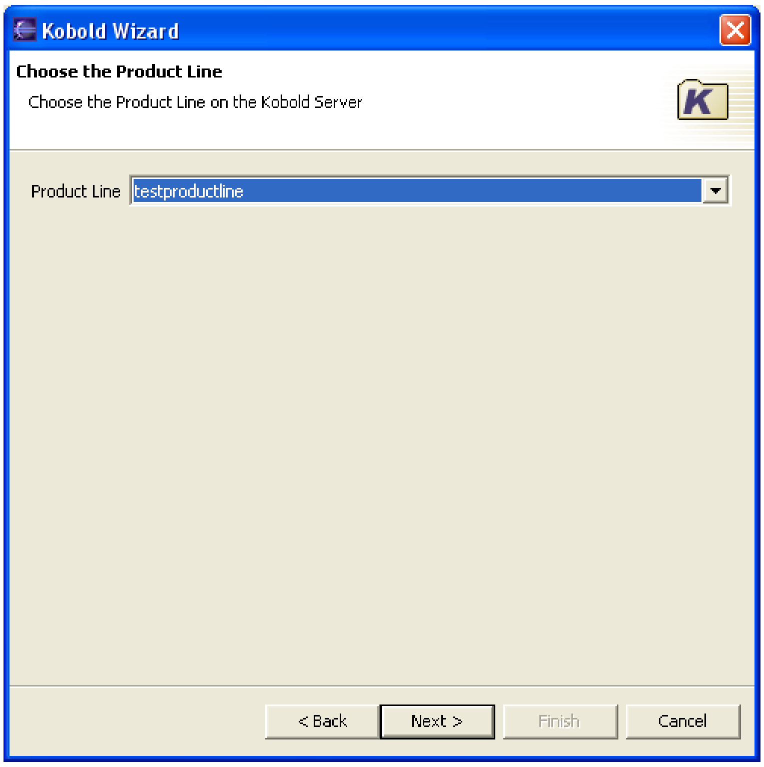
In the pallete select the ”variant” item. Click within componentA in the Architecture Editor. A variant is inserted and a dialog opens where you can enter ’variantA1’ as name of the variant (see 5.9).

Repeat these steps another two times in order to create the variants ’variantA2’ and ’variantB’. After this your architecture should look like the following (see 5.10):

5.8 Creating releases

In the pallete select the ”release” item. Click within variantA1 in the Architecture Editor. A release is inserted and a dialog opens where you can enter ’releaseA’ as name of the release (see 5.11).

Repeat these steps another two times in order to create the releases ’releaseB1’ and ’releaseB2’. After this your architecture should look like the following (see 5.12):



Choose the Product Line

Choose the Product Line on the Kobold Server

Product Line testproductline

Figure 5.5: Kobold wizard

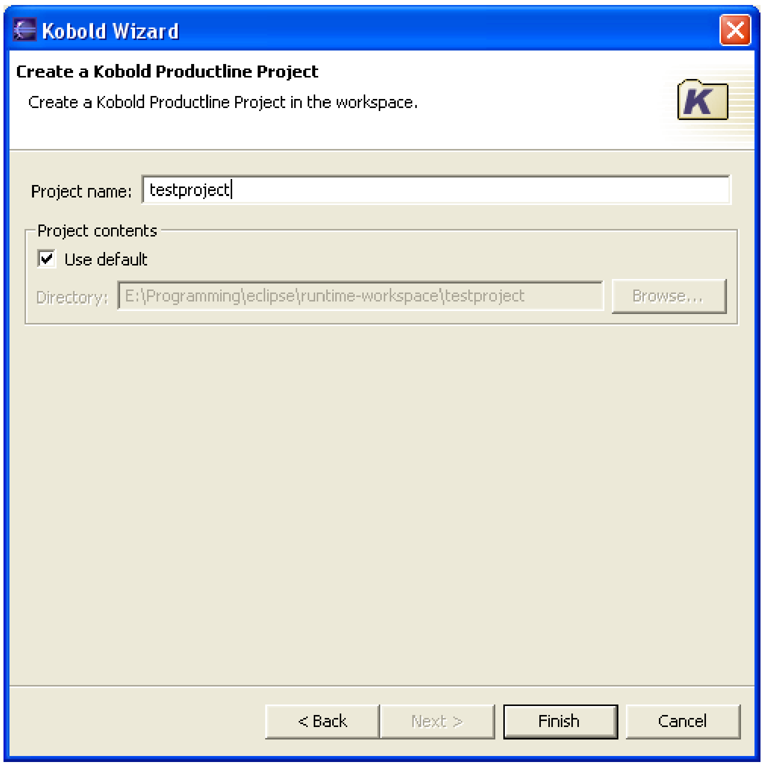
5.9 Creating dependency edges

In the pallete select the ”include edge” item. Select the assets ’variantA1’, ’releaseB1’, ’componentA’ and ’componentC’ in this order. Two dependency edges appear between the assets. After this your architecture should look like the following (see 5.13):

5.10 Creating a product

In the palette press ’compose product’. The components, etc. in the architecture editor turn grey. Select ’releaseA’. The chosen object and all the objects that are needed for it turn blue (see 5.14).

Press the ’create product’ button in the Architecture Editor. A dialog opens where you can enter ’testproduct’ as the name of your new product. After confirming the dialog, you can see your new product in the Architecture Tree and Architecture Editor (see 5.15).



Create a Kobold Productline Project

Create a Kobold Productline Project in the workspace.

Project name: testproject

Project contents

☑ Use default

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

Figure 5.6: Kobold wizard

5.11 Writing a mail

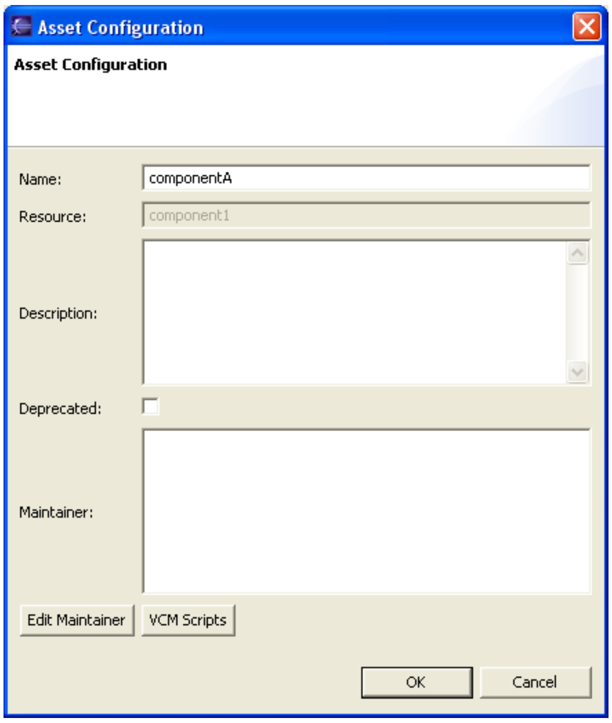
In the menu of the Workflow View select ”new mail”. A Workflow window opens where you enter ’testuser’ as the recipient in order to send yourself a testmail. Choose ’test’ as the subject of your message and enter ’this is a testmail’. Press the ’transmit’ button to send your mail (see 5.16).

Afterwards open the menu of the Workflow View and select ’Fetch messages’. Your sent message arrives in your Workflow View.

5.12 Answering your mail

In the Workflow View double-click on your testmail. The Workflow window opens where you can see the message text of the mail. Below you enter the subject ’reply’ and the text of your reply. Send the answer by pressing the ’Transmit’ button (see 5.17).

Afterwards open the menu of the Workflow View and select ’Fetch messages’. Your sent message arrives in your Workflow View.



Asset Configuration

Asset Configuration

Name: componentA

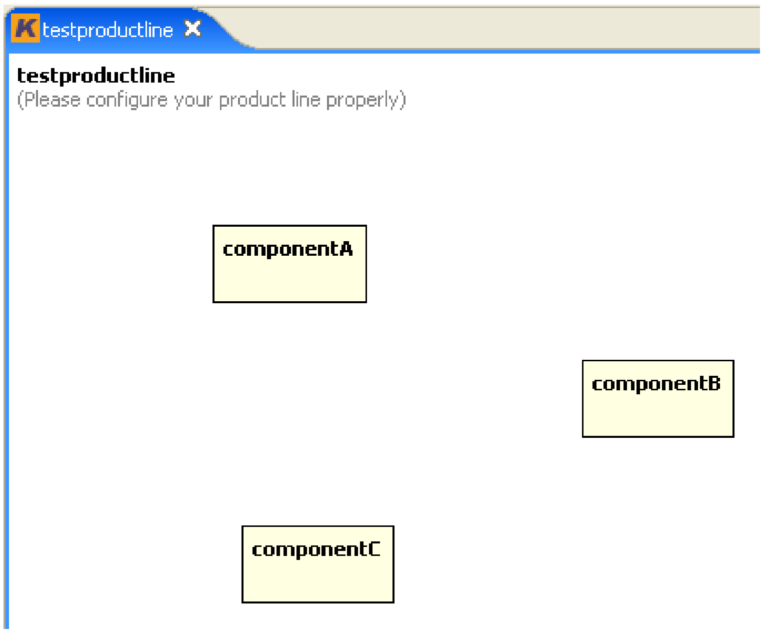
Resource: component1

Description:

Deprecated: ☐

Maintainer:

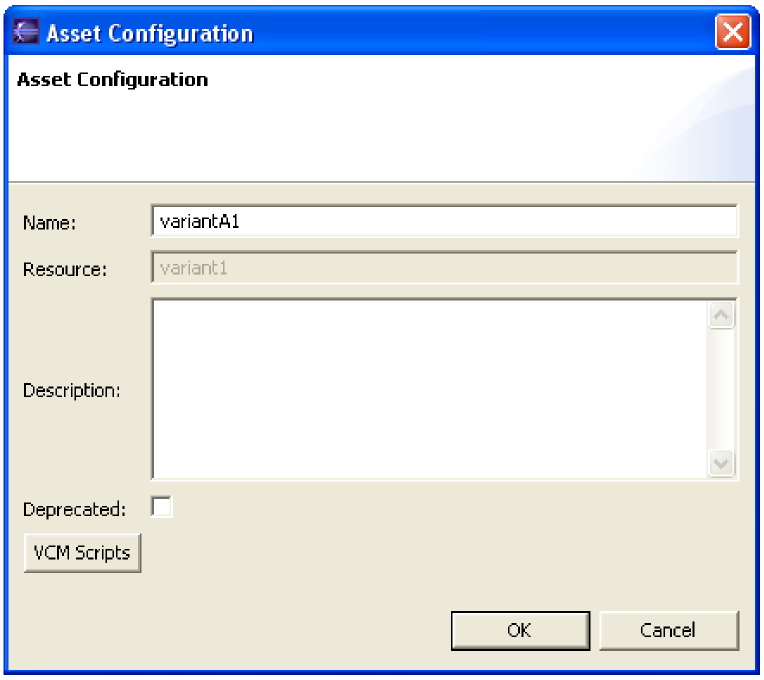
Figure 5.7: Asset Configuration Core Asset



testproductline

testproductline
(Please configure your product line properly)

Figure 5.8: Your Core Assets



Name: variantA1
Resource: variant1
Deprecated: ☐

Figure 5.9: Asset Configuration Variant

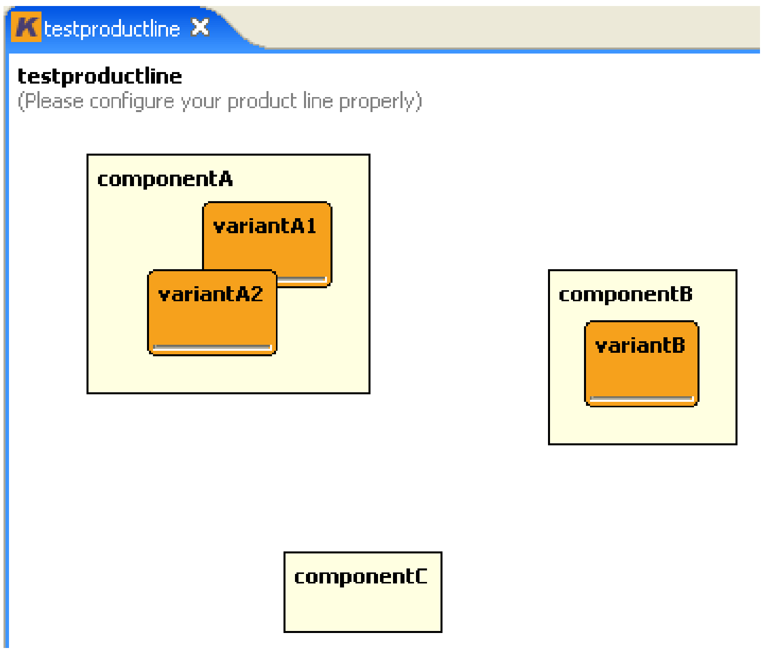
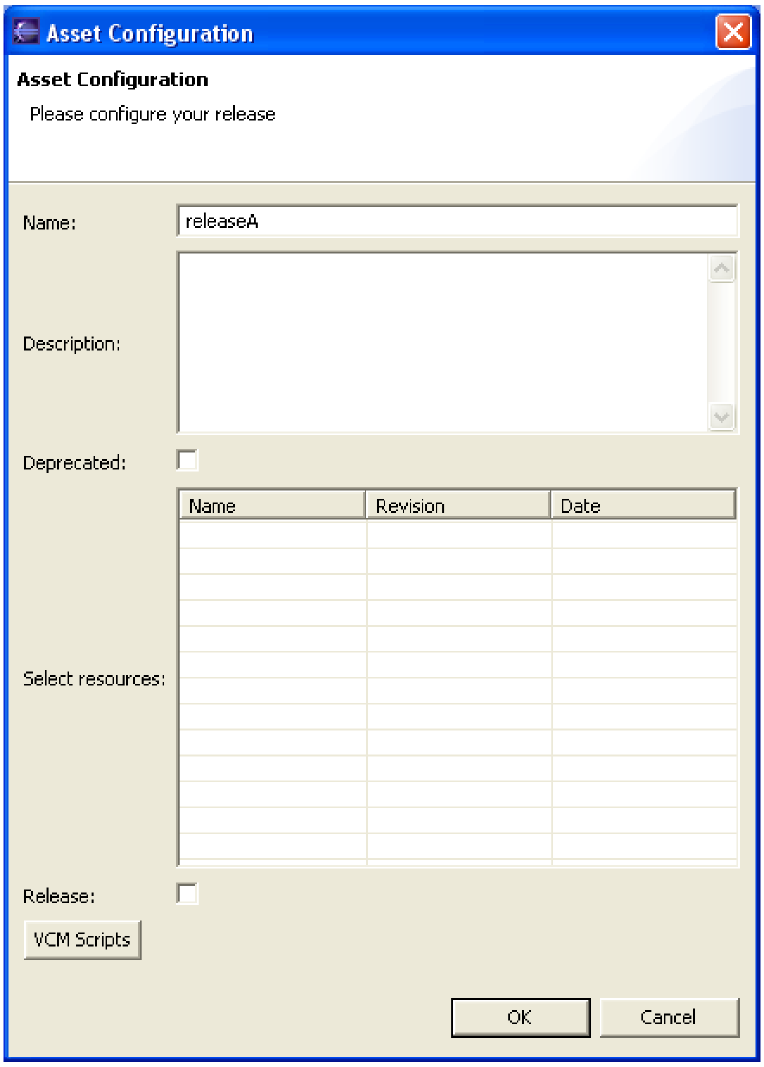


Figure 5.10: Your Variants



Asset Configuration

Please configure your release

Name: releaseA

Description:

Deprecated: ☐

Name	Revision	Date

Select resources:

Release: ☐

Figure 5.11: Asset Configuration Release

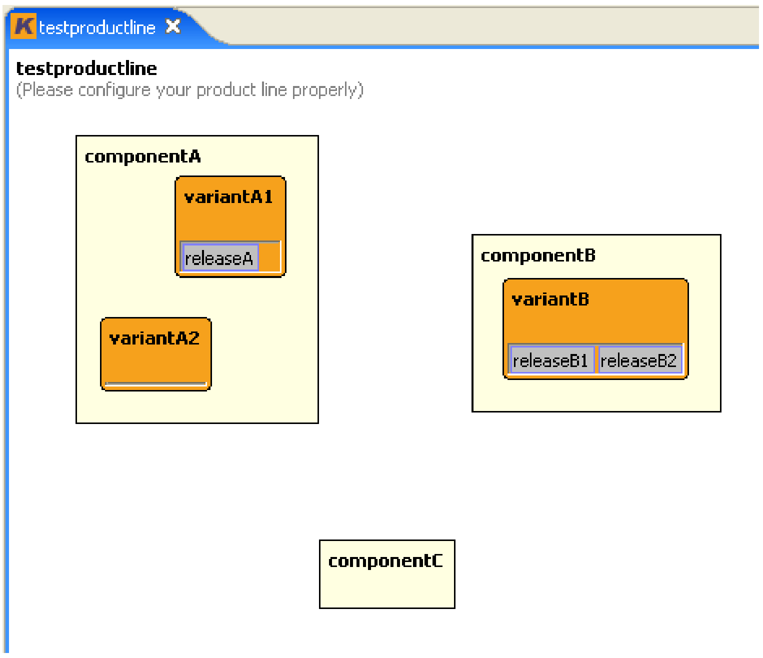


Figure 5.12: Your Releases

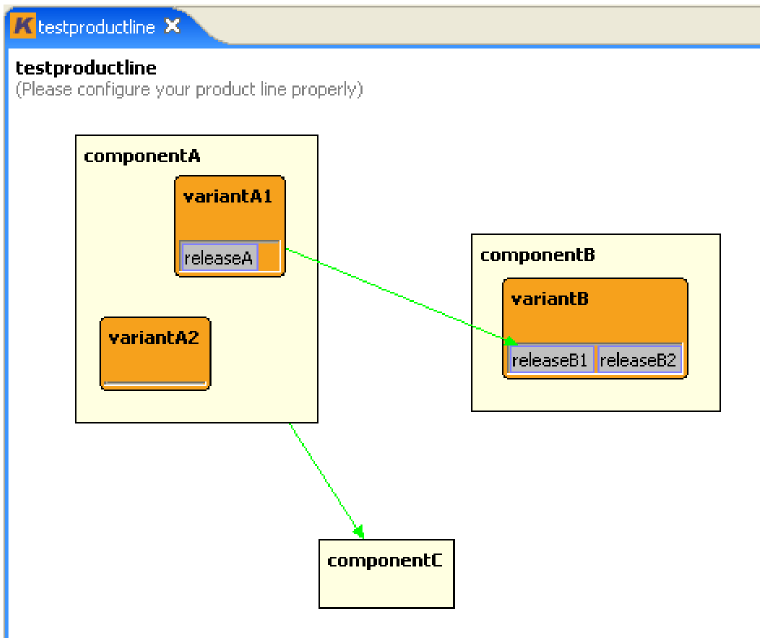


Figure 5.13: Dependency edges

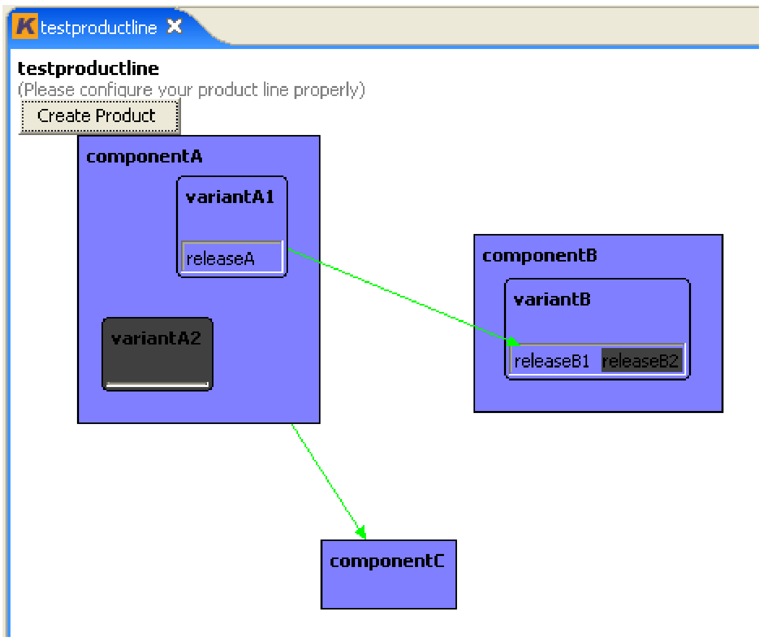
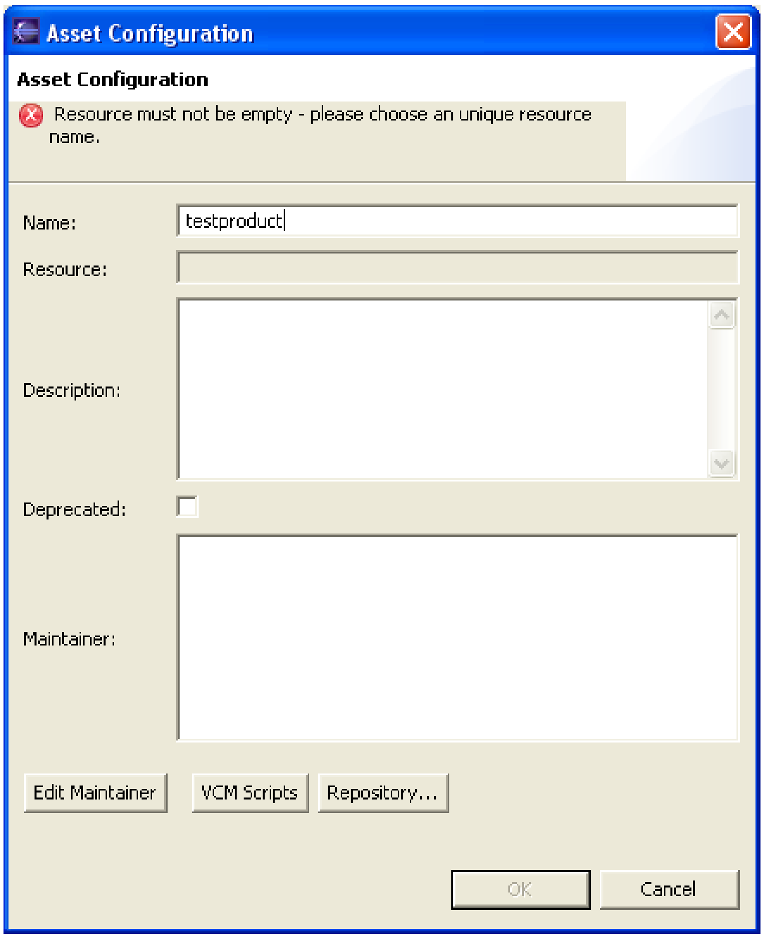


Figure 5.14: Creating a product - step 1



Asset Configuration

Asset Configuration

Resource must not be empty - please choose an unique resource name.

Name: testproduct

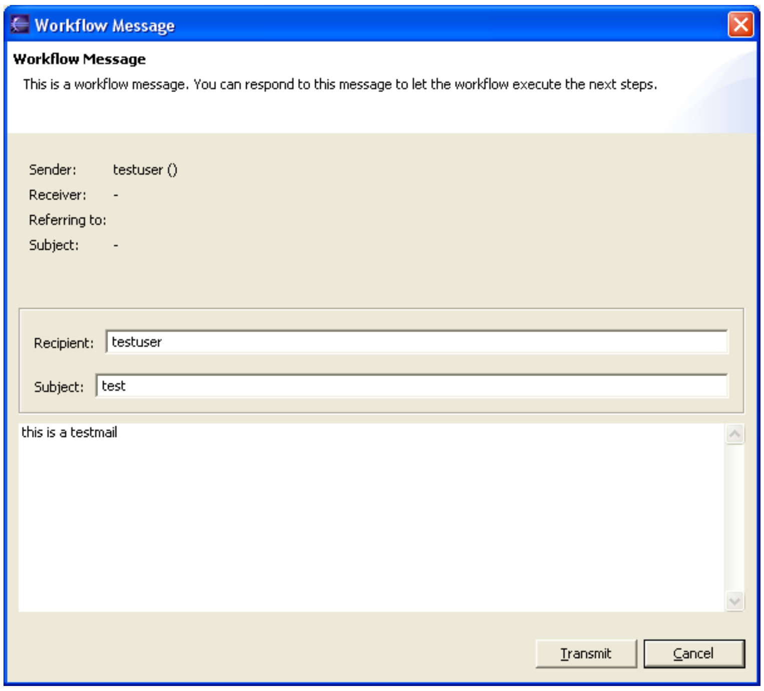
Resource:

Description:

Deprecated:

Maintainer:

Figure 5.15: Creating a product - step 2

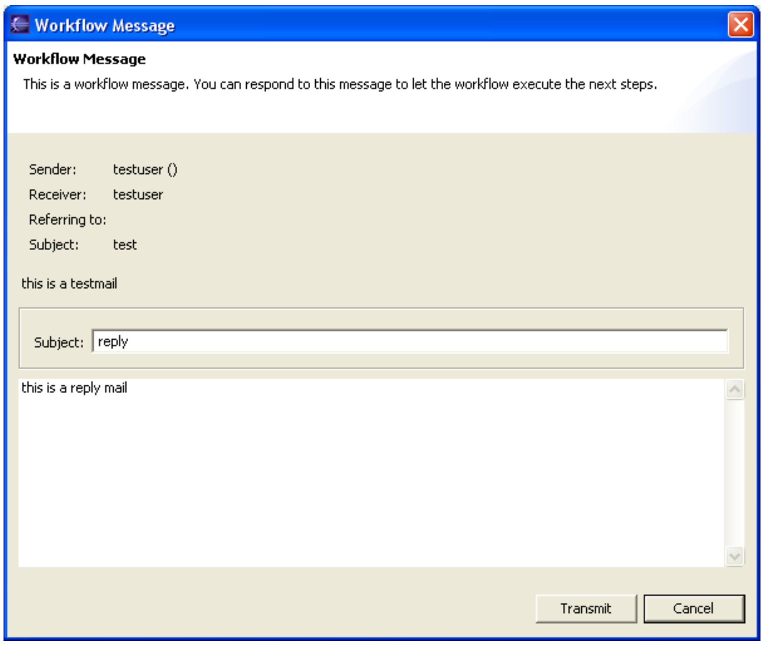

Workflow Message
Workflow Message
This is a workflow message. You can respond to this message to let the workflow execute the next steps.
Transmit
Cancel

Sender: testuser ()
Receiver: -
Referring to:
Subject: -

Recipient: testuser
Subject: test

this is a testmail

Figure 5.16: Write a mail


Workflow Message
Workflow Message
This is a workflow message. You can respond to this message to let the workflow execute the next steps.
Transmit
Cancel

Sender: testuser ()
Receiver: testuser
Referring to:
Subject: test

this is a testmail

Subject: reply

this is a reply mail

Figure 5.17: Answer a mail

6 Source Code

6.1 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);
}
```

6.2 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 1.7 2004/09/23 13:43:17 vanto Exp $
 *
 */
package kobold.common.data;

import java.text.DateFormat;
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;
private String productline = "";
private String role = "";

/**
 * 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);
}

    public String getProductline() {
        return productline;
    }
    public void setProductline(String productline) {
        this.productline = productline;
    }
    public String getRole() {
        return role;
    }
    public void setRole(String role) {
        this.role = role;
    }
```

```
/**
 * 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);
xmsg.addAttribute("productline", productline);
xmsg.addAttribute("role", role);

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");
productline = data.attributeValue("productline");
role = data.attributeValue("role");

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;
}
}
```

6.3 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 1.7 2004/08/10 10:31:19 neccaino 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);
}

    /**
     * @param username username to check for being assigned maintainer
     * @return true if a user with the passed username is a maintainer of this
     *         asset, false otherwise
     */
    public boolean isMaintainer(String username){
        Iterator it = maintainer.iterator();

        while(it.hasNext()){
            if (((User)it.next()).getUsername().equals(username)){
                return true;
            }
        }

        return false;
    }

/**
 * 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;
    }
}
```

6.4 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;
}
}
```

6.5 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;
    }

}

}
```

6.6 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;
    }
}
```

6.7 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 1.16 2004/09/23 13:43:17 vanto Exp $
 *
 */

package kobold.common.data;

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

import kobold.common.io.RepositoryDescriptor;

import org.dom4j.Element;
```

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

private List components = new ArrayList(); //TODO: remove when finally changing

    // change this to false to test server side id mapping
    private static final boolean stickToNameMapping = true;

    private HashMap componentsMap = new HashMap();

/**
 * 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.
     * @return List containing all registered components
 */
public List getComponents() {
        // will be removed when change to id mapping can be made risklessly
        if (stickToNameMapping) {
            return getComponentsNameMapping();
        }

        List ret = new ArrayList();
        for (Iterator it = componentsMap.values().iterator(); it.hasNext();) {
            ret.add(it.next());
        }

        return ret;
}

    // will be removed when change to id mapping can be made risklessly
private List getComponentsNameMapping() {
        return components;
    }

/**
 * Adds new component to this product.
 * @param component the component.
 */
    public void addComponent(Component component) {
        // will be removed when change to id mapping can be made risklessly
        if (stickToNameMapping) {
            addComponentNameMapping(component);
            return;
        }

        componentsMap.put(component.getId(), component);
    }

    // will be removed when change to id mapping can be made risklessly
private void addComponentNameMapping(Component component) {
components.add(component);
}

/**
 * Removes an existing component from this product.
 * @param component the component.
 */
public void removeComponent(Component component) {
        // will be removed when change to id mapping can be made risklessly
if (stickToNameMapping) {
            removeComponentNameMapping(component);
            return;
        }

        componentsMap.remove(component.getId());
    }
    // will be removed when change to id mapping can be made risklessly
    private void removeComponentNameMapping(Component component) {
components.remove(component);
}

    /**
     * Returns a component by its id
     * @param id id of the component to get
     * @return the component with the specified id or null if no such component
     *         exists
     */
    public Component getComponent(String id) {
        return (Component) componentsMap.get(id);
    }

/**
 * Serializes this product.
 */
public Element serialize() {
        // will be removed when change to id mapping can be made risklessly
if (stickToNameMapping) {
            return serializeNameMapping();
        }

Element element = super.serialize();

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

return element;
}

    // will be removed when change to id mapping can be made risklessly
    public Element serializeNameMapping() {
        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) {
        // will be removed when change to id mapping can be made risklessly
        if (stickToNameMapping) {
            deserializeNameMapping(element);
            return;
        }

        super.deserialize(element);
        Element compElements = element.element("components");

        for (Iterator iterator = compElements.elementIterator(Asset.COMPONENT);
             iterator.hasNext(); )
        {
            Element elem = (Element) iterator.next();
            Component c = new Component(this, elem);
            componentsMap.put(c.getId(), c);
        }
    }


    // will be removed when change to id mapping can be made risklessly
private void deserializeNameMapping(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));
}
}
}
```

6.8 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 1.23 2004/09/23 13:43:17 vanto 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 kobold.common.io.RepositoryDescriptor;

import org.dom4j.Element;


/**
 * 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();

    // change this to false to test server side id mapping
    private static final boolean stickToNameMapping = true;

/**
 * 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) {
        // will be removed when change to id mapping can be made risklessly
        if (stickToNameMapping) {
            return addProductNameMapping(product);
        }

        // 1.) refuse adding if product’s name is already registered
        if (getProductByName(product.getName()) != null) {
            return false;
        }

        // 2.) set parent and add
        product.setParent(this);
products.put(product.getId(), product);
return true;
}

    // will be removed when change to id mapping can be made risklessly
    private boolean addProductNameMapping(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 id.
     *
 * @param productId the product’s id.
     * @return the product with the specified id or null if no product with
     *         that id exists
 */
public Product getProduct(String productId) {
    return (Product)products.get(productId);
}
```

```
    /**
     * Gets a product by its name
     *
     * @param productName name of the product to get
     * @return Product with the specified name or null if no product with that
     *         name exists
     */
    public Product getProductByName(String productName) {
        for(Iterator it = products.values().iterator(); it.hasNext();) {
            Product p = (Product) it.next();
            if (p.getName().equals(productName)) {
                return p;
            }
        }

        return null; //no registered product with that name
    }

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

    /**
     * Gets a coreasset by its name.
     *
     * @param coreAssetName name of the coreasset to get.
     * @return the coreasset with the specified name or null if no coreasset
     *         with that name exists
     */
    public Component getCoreAssetByName(String coreAssetName) {
        for(Iterator it = coreassets.values().iterator(); it.hasNext();) {
            Component c = (Component) it.next();
            if (c.getName().equals(coreAssetName)) {
                return c;
            }
        }

        return null; //no registered core asset with that name
    }
```

```
/**
     * Gets all products.
     * @return List containing every registered product
 */
public List getProducts() {
    return new ArrayList(products.values());
}

/**
 * Gets all coreassets.
     * @return List containing every registered core asset
 */
public List getCoreAssets() {
    return new ArrayList(coreassets.values());
}

/**
 * Removes a product by its id.
 * @param productId the product’s id.
     * @return the removed Product if a product with the passed id existed, null
     *         otherwise
 */
public Product removeProduct(String productId) {
Product ret = (Product) products.remove(productId);

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

        return ret;
}

    /**
     * Removes a product by its name.
     *
     * @param productName name of the product to remove
     * @return the removed Product object, or null if no product with the
     *         specified name exists
     */
    public Product removeProductByName(String productName) {
        String id = getProductIdByName(productName);
        return id == null ? null : removeProduct(id);
    }
    /**
     * @param productName name of the product whose id should be returned
     * @return id of the product with the specified name or null if no such
     *         product exists
     */
    public String getProductIdByName(String productName) {
        Product p = getProductByName(productName);
        return p == null ? null : p.getId();
    }

/**
 * 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) {
        // will be removed when change to id mapping can be made risklessly
        if (stickToNameMapping) {
            return addCoreAssetNameMapping(coreasset);
        }

        // 1.) refuse adding if coreasset’s name is already registered
        if (getCoreAssetByName(coreasset.getName()) != null) {
            return false;
        }

        // 2.) set parent and add
        coreasset.setParent(this);
        coreassets.put(coreasset.getId(), coreasset);
        return true;
}

    // will be removed when change to id mapping can be made risklessly
    private boolean addCoreAssetNameMapping(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 by its id.
 * @param coreasset the coreassset to remove.
     * @return the removed coreasset if a core asset with the specified id
     *         existed, null otherwise
 */
public Component removeCoreAsset(String coreAssetId) {
Component ret = (Component) coreassets.remove(coreAssetId);

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

        return ret;
}

    /**
     * Removes a core asset by its name.
     *
     * @param coreAssetName name of the core asset to remove
     * @return the removed core asset or null if no core asset with the
     *         specified name exists
     */
    public Component removeCoreAssetByName(String coreAssetName) {
        String id = getCoreAssetIdByName(coreAssetName);
        return id == null ? null : removeCoreAsset(id);
    }

    /**
     * @param coreAssetName name of the core asset whose id should be returned
     * @return id of the core asset with the specified name or null if no such
     *         product exists
     */
    public String getCoreAssetIdByName(String coreAssetName) {
        Component c = getCoreAssetByName(coreAssetName);
        return c == null ? null : c.getId();
    }

/**
 * 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) {
        // will be removed when change to id mapping can be made risklessly
        if (stickToNameMapping) {
            deserializeNameMapping(element);
            return;
        }

super.deserialize(element);
    Element coreassetElements = element.element("coreassets");
for (Iterator iterator = coreassetElements.elementIterator(Asset.COMPONENT);
 iterator.hasNext(); )
{
Element elem = (Element) iterator.next();
addCoreAsset(new Component(this, elem));
}

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

    // will be removed when change to id mapping can be made risklessly
    private void deserializeNameMapping(Element element) {
        super.deserialize(element);
        Element coreassetElements = element.element("coreassets");
        for (Iterator iterator = coreassetElements.elementIterator(Asset.COMPONE
             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);
        }
    }
}
```

6.9 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;
}
}
```

6.10 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();
    }
```

```
}
```

6.11 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 representing 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;
    }
}
```

6.12 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;
}

}
```

6.13 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 1.26 2004/09/23 13:43:17 vanto Exp $
 *
 */
package kobold.common.data;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Set;

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;
}
}
```