Rhapsody operational concept

IBM® Rational® Rhapsody® StartUp Training

WILLERT.
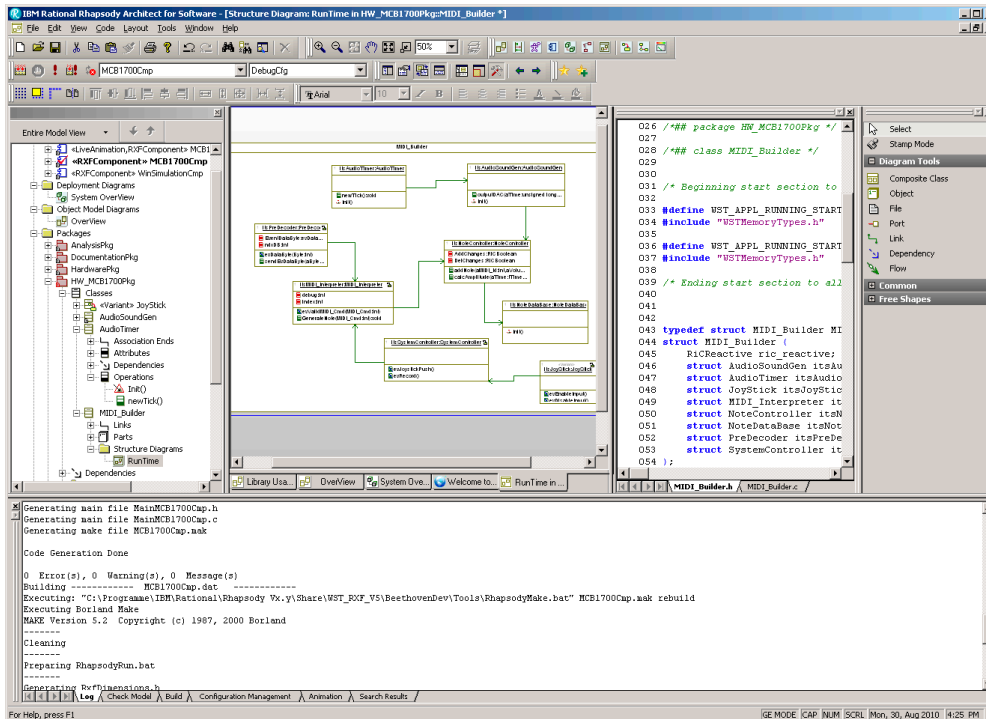
become familiar with Rhapsody

Rhapsody belongs to the category of model driven software development tools. Mostly, these tools make use of modeling languages like UML and SysML to create software models or abstractions. They can be divided into two major groups. Firstly, into pure modeling tools like the Enterprise Architect from Sparx Systems. Such tools are often used for documentation purposes.

The other group also offers the functionality of generating code from models. So there is no separation between software and documentation. Baring this in mind, we don't have to waste working hours, keeping software and documentation consistent. This is done automatically. Fortunately, IBM Rational Rhapsody belongs to the second group. This saves us having to use other documentation tools and gives us more time for programming ;-)

The integration of IBM Rational DOORS is another great advantage. DOORS is a powerful requirement management tool that can integrated in Rhapsody seamlessly. So now, we have a complete tool chain. In summary we have a requirements management tool, a software engineering tool and a complete documentation which forms our product.
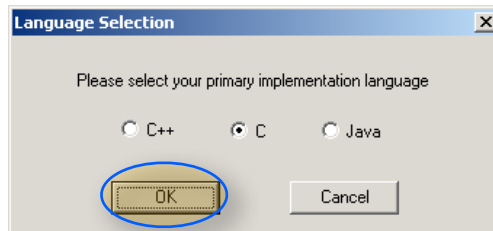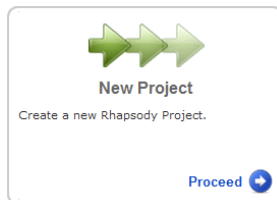
In Rhapsody, the target language is changeable. For our embedded systems, we are only interested in ANSI C. But Rhapsody can also generate Java, C++ or C# code.

In order to work with Rhapsody efficiently, it is important to understand the basic functionalities. First of all, Rhapsody is more complex than MS WORD and has a lot of special features. Fortunately, we need only a small number.
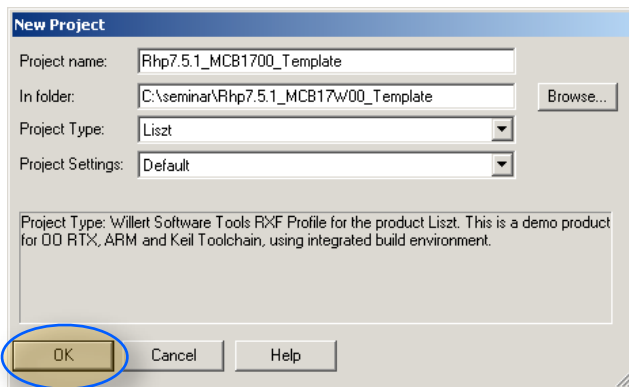
Rhapsody Basics

create a project

Select File / New from the main menu, or click the "New Project" Button in the welcome screen

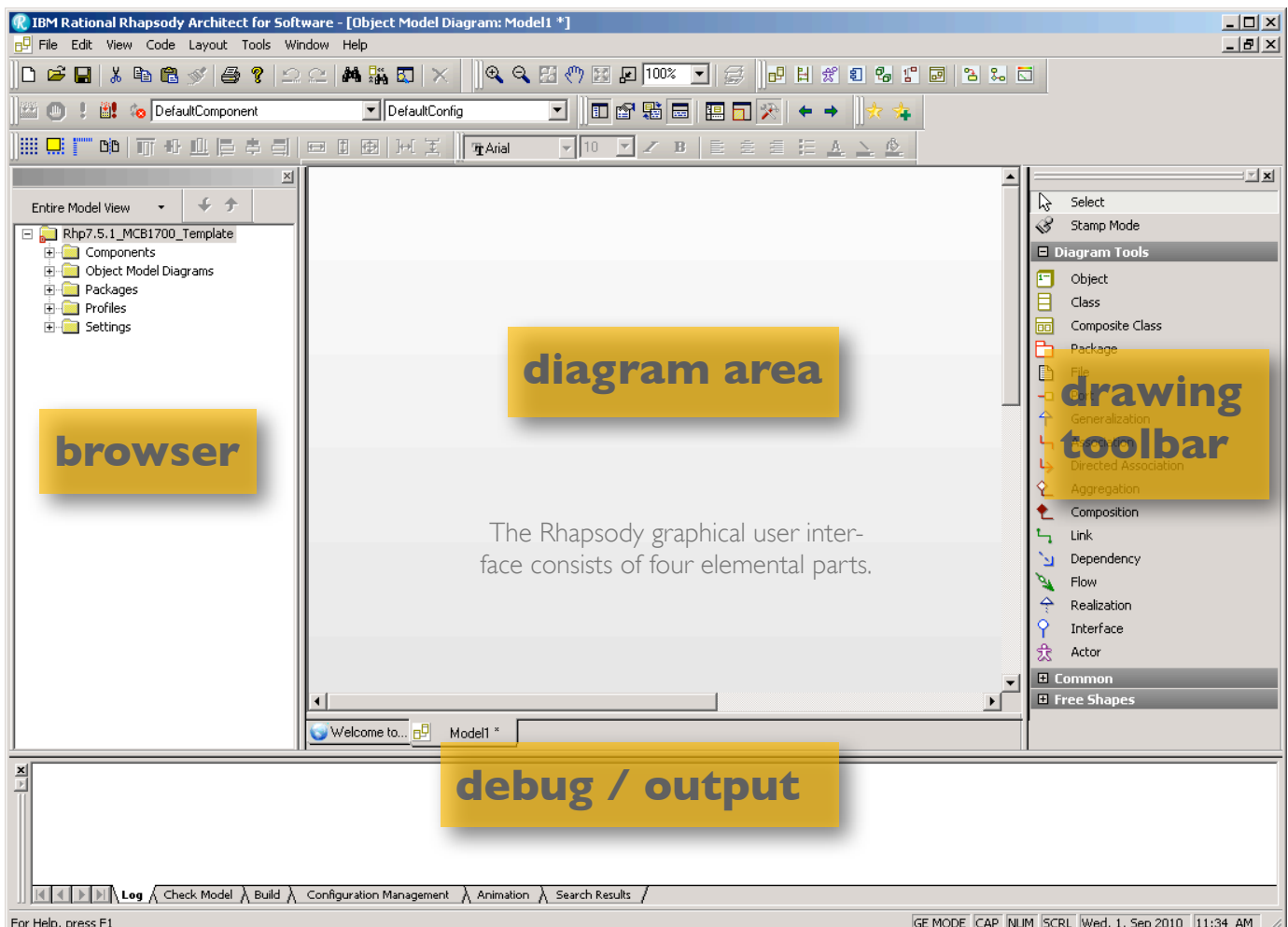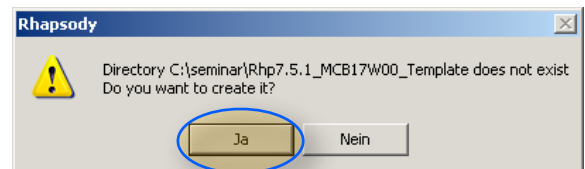**New Project**
Create a new Rhapsody Project.

Proceed ➜

Language Selection

Please select your primary implementation language

○ C++   ◉ C   ○ Java

[OK]   [Cancel]

Rational Rhapsody Architect for Software supports three different languages. C is already preselected, thats fine.

New Project

Project name:	Rhp7.5.1_MCB1700_Template
In folder:	C:\seminar\Rhp7.5.1_MCB17W00_Template
Project Type:	Liszt
Project Settings:	Default

Project Type: Willert Software Tools RXF Profile for the product Liszt. This is a demo product for OO RTX, ARM and Keil Toolchain, using integrated build environment.

[OK]   [Cancel]   [Help]

In the next window we are asked to check some basic settings. Select a project folder

Project name:	`Rhp7.5.1_MCB1700_Template`
In Folder:	`C:\where\ever\you\want`
Project Type:	`Liszt`
Project Settings:	`Default`

Rhapsody

⚠ Directory C:\seminar\Rhp7.5.1_MCB17W00_Template does not exist
Do you want to create it?

[Ja]   [Nein]

IBM Rational Rhapsody Architect for Software - [Object Model Diagram: Model1 *]

File Edit View Code Layout Tools Window Help

browser

Entire Model View
- Rhp7.5.1_MCB1700_Template
  - Components
  - Object Model Diagrams
  - Packages
  - Profiles
  - Settings

diagram area

The Rhapsody graphical user interface consists of four elemental parts.

drawing toolbar

Select
Stamp Mode
Diagram Tools
- Object
- Class
- Composite Class
- Package
- File
- Generalization
- Directed Association
- Aggregation
- Composition
- Link
- Dependency
- Flow
- Realization
- Interface
- Actor
- Common
- Free Shapes

Welcome to...   Model1 *

debug / output

Log Check Model Build Configuration Management Animation Search Results

For Help, press F1

GE MODE CAP NUM SCRL Wed, 1, Sep 2010 11:34 AM

browser (and his elements)

First of all the model browser. It works like the windows explorer. You can use the +/- buttons, to navigate in and out. All project elements are accessible through the browser. There are five model elements worth mentioning.
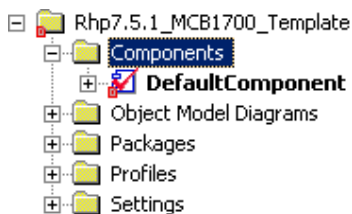
Components

Here you can add external source- and header-files or libraries to your project. You also have the opportunity, to specify which model elements should be included by the component. For example, if you would like to build a project for two different targets (e.g. PC and Microcontroller), i suggest that each target should have its own components.
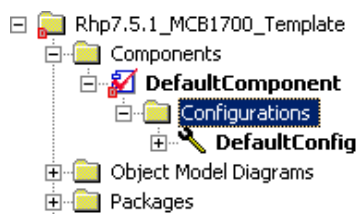
Configurations

A configuration is a sub item of a component. As with the components, you also have the opportunity to implement your own code. Every configuration has its own environmental settings, where you can choose a compiler, or modify compiler- and link-switches. We will use the configuration to create different Buildsets (debug / release) for a component.

Object Model Diagrams

A fresh Rhapsody project already has a UML diagram inside. It is an OMD (Object Model Diagram). This diagram is associated with the default package. All UML elements you add here, will also appear in the standard package automatically.
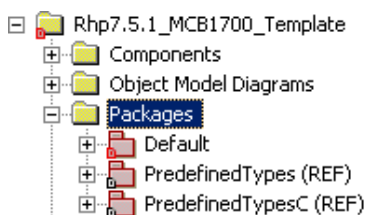
Packages

Packages are containers for UML stuff and other packages. With packages, you have the possibility, to divide your project into smaller parts. Of course, it is not necessary to do so. Additionally you can also gather all project elements in one package. But, this is not best practice for large projects. Use nested packages to create subsystems or equivalent packages to divide your system.
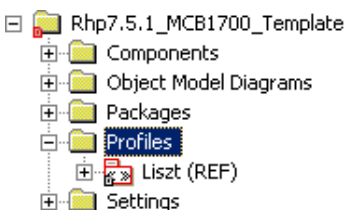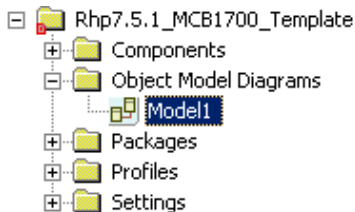
Profiles

Profiles give you the chance to apply various properties to your project using a single file. Therefor, when creating a new project, you don't have run through all settings again manually. All you need to do, is to load a single profile.

diagram area & drawing toolbar

As the name suggests, all possible diagram types will be edited in this area. The diagram area consists of a large painting surface, a tab bar and and the layout-, zoom-toolbar. The diagram area can display multiple diagrams at once. You can use the tab bar to switch between them. It is like the tab function in your web browser. It's actually pretty simple. The drawing toolbar of the diagram area is located to the right hand side. It shows all possible UML elements that fit your selected diagram type. You can open a diagram by double clicking it in your model browser.

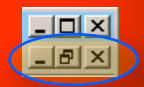Make a double click on your Object Model Diagram "Model1".

ATTENTION:

If you want to close a diagram, use the windows close button in the upper right corner of your rhapsody window. But take the second button bar. The first one is for the rhapsody window.

Now there are two elements in the your tab bar...

close the Welcome screen, but...

We will now draw some UML items and get familiar with the layout functionality. Select a class from the diagram toolbar and left click into the diagram surface.
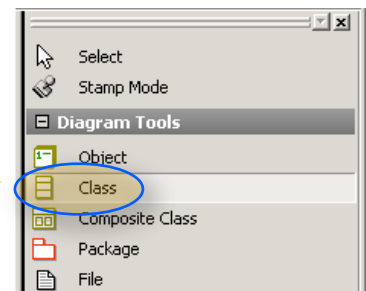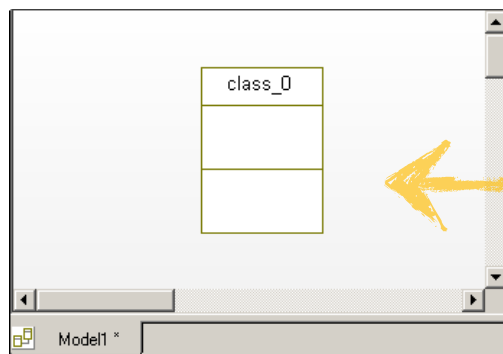
Congratulations, you have just created a class. Lets go back to the toolbar once again, we can see that the class is not selected anymore. So if we need to draw more than one class in a single step, we have to activate the stamp mode. Click on "Stamp Mode", select the Class icon again and draw some more classes in your diagram.
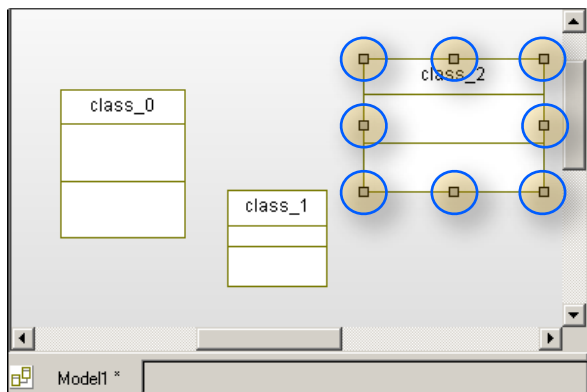
If it is necessary we can apply attributes and methods to our classes afterwards. These can be also displayed in the classbox. If you would like to see them, it could be helpful to change the dimensions of the class. Click on a class, and you will see anchor points with which you can change the size of the class.

Sometimes you have to change the dimensions of more than one class simultaneously. Hold down the "ctrl - key" and select more than two classes. Look at the layout toolbar. As you can see, a couple of icons have now become active.
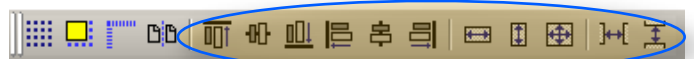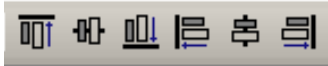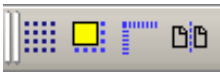
With these buttons we can align objects to the edges of the diagram. Note, that the last selected object serves as a reference.

With the first three buttons, we can adjust the size of several objects. With the last two we can set uniform distances between objects. The last selected object is used again as reference.

Here we have the opportunity to display a grid in the diagram. We can also align objects to that grid. If you like rulers, you can also make this shown. The last button is useful if you want to print your diagram. After the diagram printing dialogue, you can display page margins in your diagram.


With the zoom toolbar, you can fit your diagram to a correct size. Use the magnifier to zoom in or out. "Zoom to fit" is also very useful, it fits the entire content of a diagram into a single view.

Deleting model elements

We have just created a few classes in the Object Model Diagram. Since our browser contains all the objects of our project, these classes must also appear in the browser. Take a look at the default package. You will find the diagram elements here too.

In the contextual menu from a class in the OMD, you will see two methods for deleting:
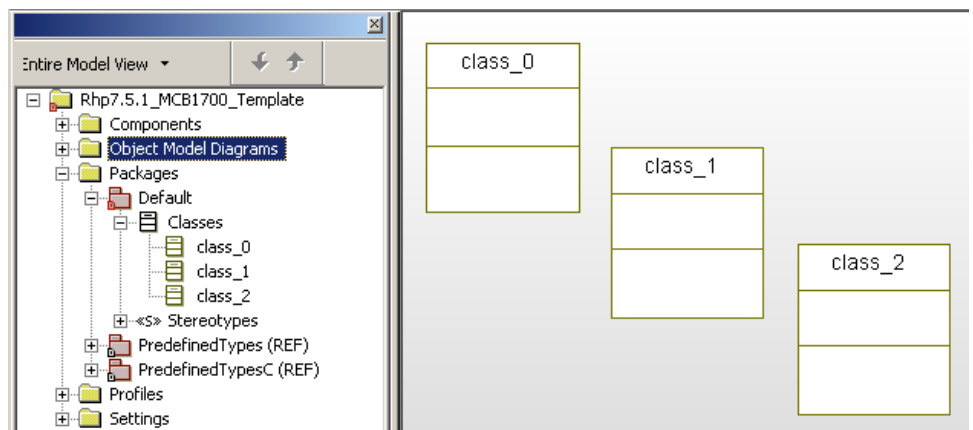
Delete from Model
is deleting the class from your project completely.

Remove from View
removes the class only from your diagram view. The class is also part of the project.

Attention:

If you highlight an object in a diagram, the delete-key on your keyboard produces a delete from the diagram view. Highlight an object in the browser and the delete-key produces a delete from model.

debug / output

The debug / output window shows us occurring errors and warnings from the generating and building process.
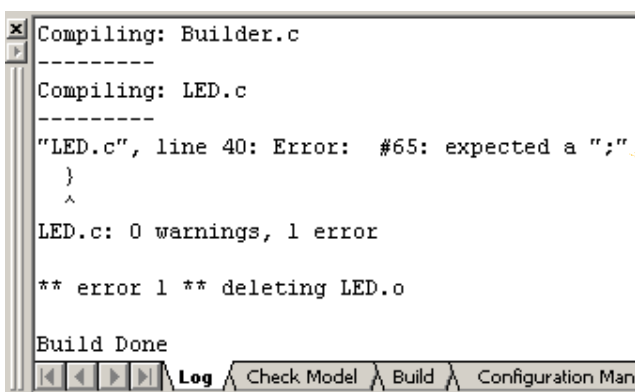
When an error occurs during the process, click on the error message and rhapsody will show you the location of the bug in the project automatically. This is just an example, we use this function later.



```
34  /*## operation Init() */
35  void LED_Init(LED* const me) {
36      /*#[ operation Init() */
37          GLCD_Init();
38          GLCD_Clear(Red)
39      /*#]*/
40  }
```

Rhapsody Specials

Now, we are familiar with the most basic rhapsody functions, it is time for some special things and features. With a few settings, efficiency can improve dramatically.

Customize your rhapsody window

Should your pc allow a high resolution, there are two more extra subwindows for our rhapsody window. Take a look at the window-toolbar.

S/H Browser – S/H Features – S/H Active Code View – S/H Output Window – Toggle Arrange Options
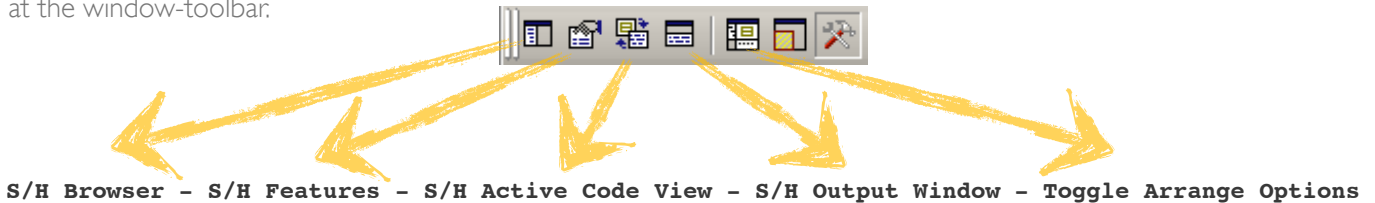
Show/Hide Browser is preselected. We need the browser every time. There is no need to remove the browser from the view.

Show/Hide Features shows us the features window from a selected project item. A useless button, because a double click on a project item does the same.

Show/Hide Active Code View shows a preview of the generated code. One of the most important features. If it allows your resolution, you should anchor active code view in your rhapsody window. Before active code view shows us some lines of code, we have to click "Generate". Don't worry, we will do this later.

Show/Hide Output Window is preselected. Since we build and run tests quiet often, this window should be an integral part of the rhapsody window.

Toggle Arrange Options will integrate the features window in your permanent view. People with high-resolution screens should anchor this subwindow, because we will spend some time here.

The Features Window

If we edit the properties of an project element, we always do this using the features window. Open this window by double clicking an project element. Try this with a testclass from page 4. Now click on another project element e.g. the object model diagram or something else. You will see that the features window is context sensitive. But...
You can also pin a features window in his context. Use the button in the upper left corner for this action. If you have pinned a features window, you can open other features windows double clicking on other project elements.
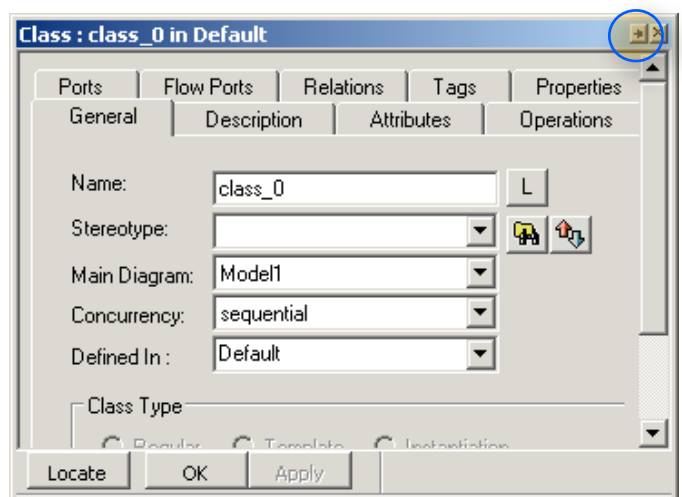
Display line numbers

Line numbers are a great thing. If you want line numbers to be displayed in your code areas, make a right click on a code area e.g. Active Code View and select properties. In the misc section, you can switch line numbering globally on / off. Feel free in your choice of numeral systems.
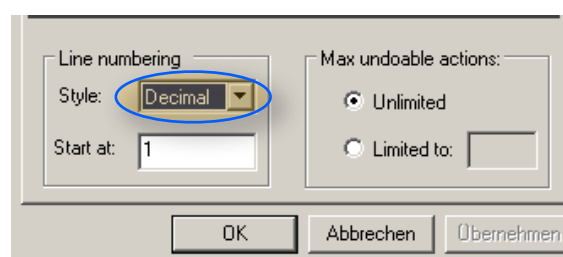
Coding Guidelines

You have probably noticed that the names of several elements have certain extensions. This is a coding guideline we use. You do not have to adapt it, but you will see that it makes life easier. You can also design your own code.

```
evXX      event
opXX      triggered operation
xxxOMD    Object Model Diagram
xxxCMP    Component
xxxPkg    Package
C_xxx     Class
```
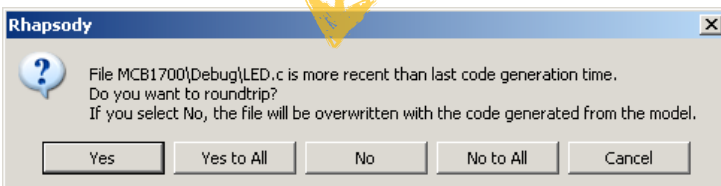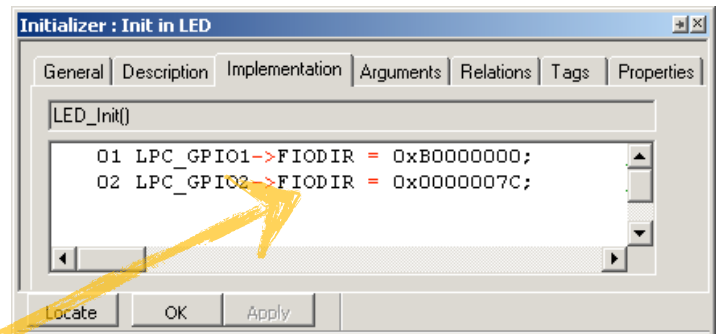
Roundtripping

Rhapsody will generate code from your UML project. You can integrate your own code into this framework. That is the normal way. But you can also make changes in the generated code. Rhapsody will ask you to update the model next time you generate code. We will try this later.

```
35  void LED_Init(LED* const me) {
36      /*#[ operation Init() */
37      LPC_GPIO1->FIODIR = 0xB0000000;
38      LPC_GPIO2->FIODIR = 0x0000007C;
39      /*#]*/
40  }
```
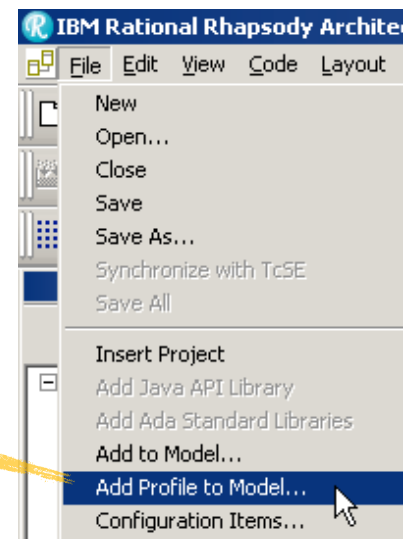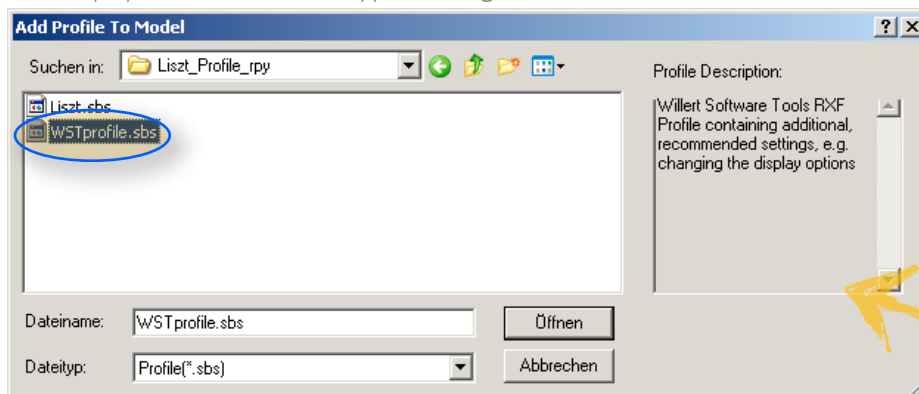

Initializer : Init in LED

General | Description | Implementation | Arguments | Relations | Tags | Properties

LED_Init()

```
01 LPC_GPIO1->FIODIR = 0xB0000000;
02 LPC_GPIO2->FIODIR = 0x0000007C;
```

Locate   OK   Apply

Rhapsody

File MCB1700\Debug\LED.c is more recent than last code generation time.
Do you want to roundtrip?
If you select No, the file will be overwritten with the code generated from the model.

Yes | Yes to All | No | No to All | Cancel

A t t e n t i o n :

Normally, roundtripping works very well, but beware not everything can be round-tripped.

Adding a new Profile

Now, we add a new profile to our project. The WSTprofile.sbs is an optional profile which sets several properties recommended by Willert Software Tools. It will modify some graphical settings such as: rectilinear arrows for statecharts and display names and stereotypes in diagram views.

Add Profile To Model

Suchen in: Liszt_Profile_rpy

Liszt.sbs
WSTprofile.sbs

Profile Description:

Willert Software Tools RXF Profile containing additional, recommended settings, e.g. changing the display options

Dateiname: WSTprofile.sbs   Öffnen
Dateityp: Profile(*.sbs)   Abbrechen

IBM Rational Rhapsody Archite...

File   Edit   View   Code   Layout

New
Open...
Close
Save
Save As...
Synchronize with TcSE
Save All

Insert Project
Add Java API Library
Add Ada Standard Libraries
Add to Model...
Add Profile to Model...
Configuration Items...

```
-> Add Profile to Model
...Rhapsody\Share\WST_RXF_V5\LisztVx\Config\Profiles\Liszt_Profile_rpy\WSTprofile.sbs
```

As a result, you will see the WSTprofile.sbs in your model browser.

```
Packages
Profiles
    Liszt (REF)
    WSTprofile (REF)
Settings
```

Product:

Embedded UML Start-Up Training

Author:

Marco Matuschek

Editor:

Willert Software Tools GmbH

Hannoversche Strasse 21
DE - 31675 Bückeburg

www.willert.de
info@willert.de