

RiA in Eclipse

I Introduction

This document describes how to install RiA in Eclipse and how to use it.
RiA in Eclipse has been tested only with GNAT compiler.

II Requirements

To be able to run RiA on Eclipse, the following tools must be installed. The install is based on GNATBench which enables to have an Ada perspective, and create an IDE project to be connected to the Rhapsody configuration.

- Eclipse 3.4.2
- Rhp 7.5.1 or greater
- Eclipse/CDT
- GNATBench 2.3.0
- Make.exe for GNAT

III Install

To install RiA and GNATbench into Eclipse several steps must be followed in the correct order.

- Install Eclipse 3.4.2
- Install Eclipse CDT
- Install gnat bench 2.3.0
- Install a make.exe into Gnat install
- Install Rhp

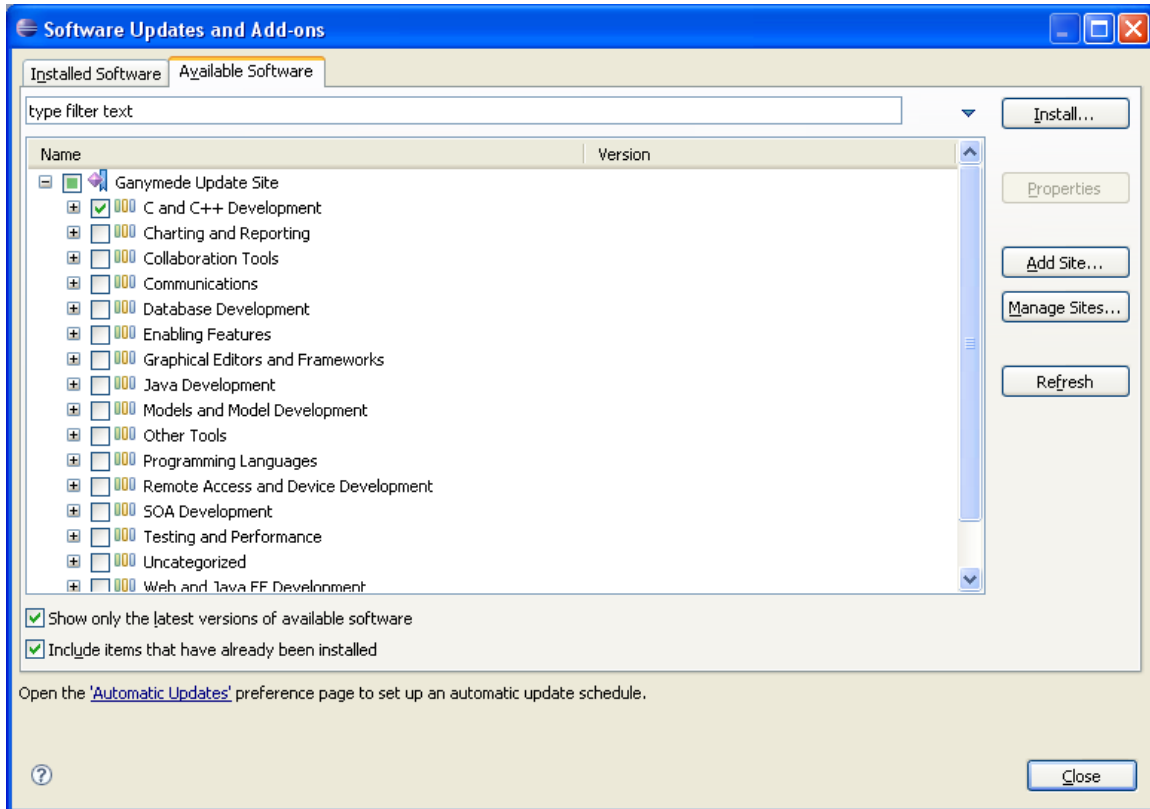
III.1 Install Eclipse 3.4.2

Install the version 3.4.2 of Eclipse.

III.2 Install Eclipse CDT

Open the install popup window of Eclipse with menu : help/install software

Install eclipse CDT first, using predefined update site



III.3 Install Gnatbench

Install GNATBench using GnatBench installer (not eclipse interface)

gnatbench-2.3.0-bin-eclipse.exe

Install 'Make' Utility

You must also install the 'make' utility before using GNATbench. This utility is available from a number of web sites.

Windows users should take the following steps to get the file from the AdaCore web site. (Other supported operating systems will very likely have 'make' already installed.) First, from the [home page](#), log in to GNAT Tracker using your account number and password.

Then select the "x86-windows" platform and the version number of the compiler you are using. Then expand the "Tools" category, under which you will see a file named "gnumake-version-number-nt.exe" that you can download. Download it, rename it to "make.exe", and place it somewhere on your path. (C:\GNATPRO\6.0.2\bin)

GNAT Tracker
The GNAT Pro customer site

Dashboard Tickets **Download** Wavefronts Documentation Developer Intern Program

Platform: x86 Windows (32 bits) ▼

Click ► to select sources or individual files

Development and Testing

- ☐ ► **GNAT Pro Ada** 6.0.2 ▼ - Ada compiler, debugger, tools, libraries i 73 Mb
- ☐ ► **AUnit** 2.02 - Ada unit testing framework i 2 Mb
- ☐ ► **GPRbuild** 1.4.1 ▼ - automatic builds of multi-language systems i 13 Mb

IDEs

- ☐ ► **GPS** 4.4.1 ▼ - GNAT Programming Studio IDE i 23 Mb
- ☐ ► **Remote GPS** 4.4.1 ▼ - GPS IDE on Linux/Windows for any target i 94 Mb
- ☐ ► **GNATbench** 2.4.0 ▼ - Ada plugin for the Eclipse IDE i 32 Mb

Interfacing and Libraries

- ☐ ► **XMLAda** 2.3 - library to process XML streams i 718 kb
- ☐ ► **Win32 Ada** 6.0.2 - Ada API to the Windows library i 2 Mb

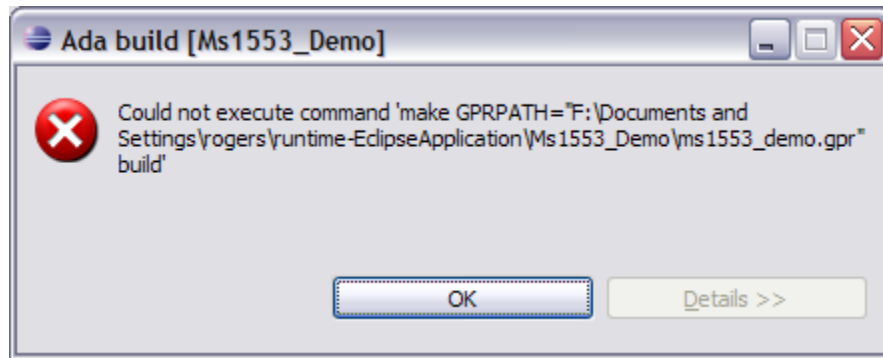
Miscellaneous

- ☒ ▼ **Utils** 6.0.2 - utilities i 121 kb
 - ☒ [gnumake-3.79.1-nt.exe](#) Jul 02, 2007 121 kb
 - Sources

Package files (with checksums) in an archive: ☐ tar ☒ zip

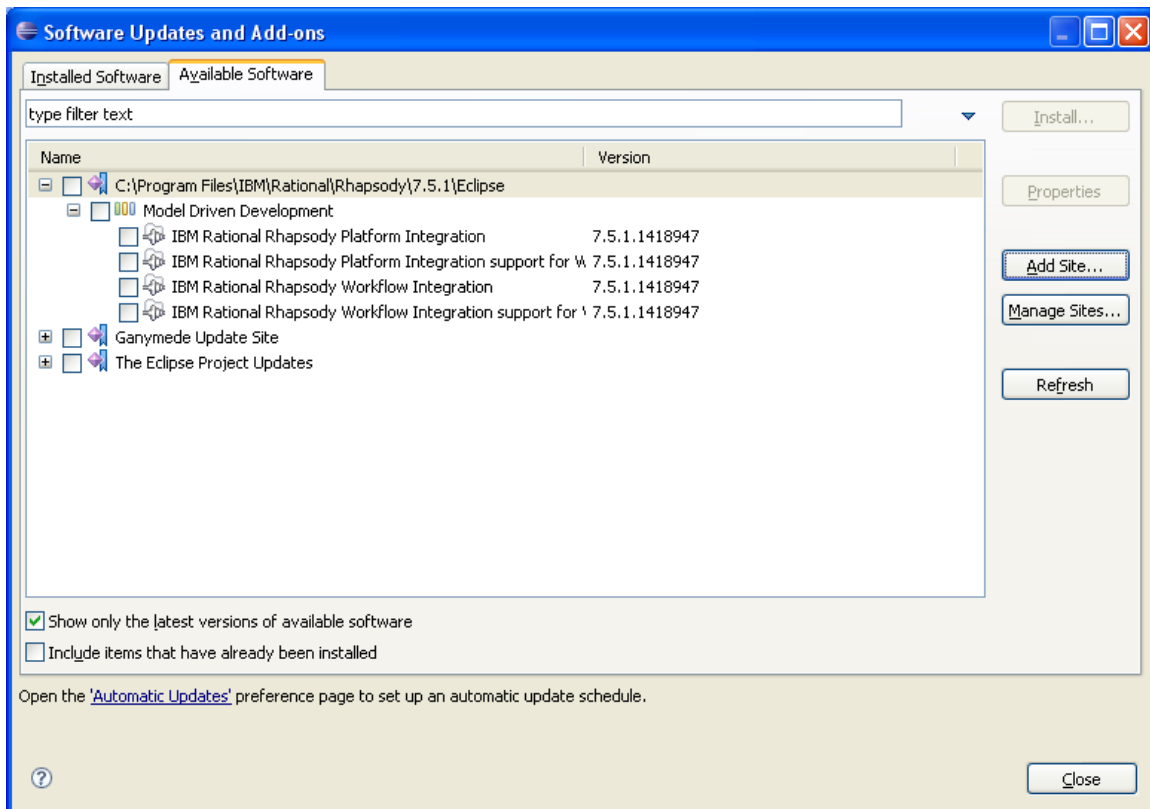
[Download Selected Files](#)

If you do not have a 'make' utility installed you will get an error popup dialog similar to the one below when attempting to use the compiler and associated tools:

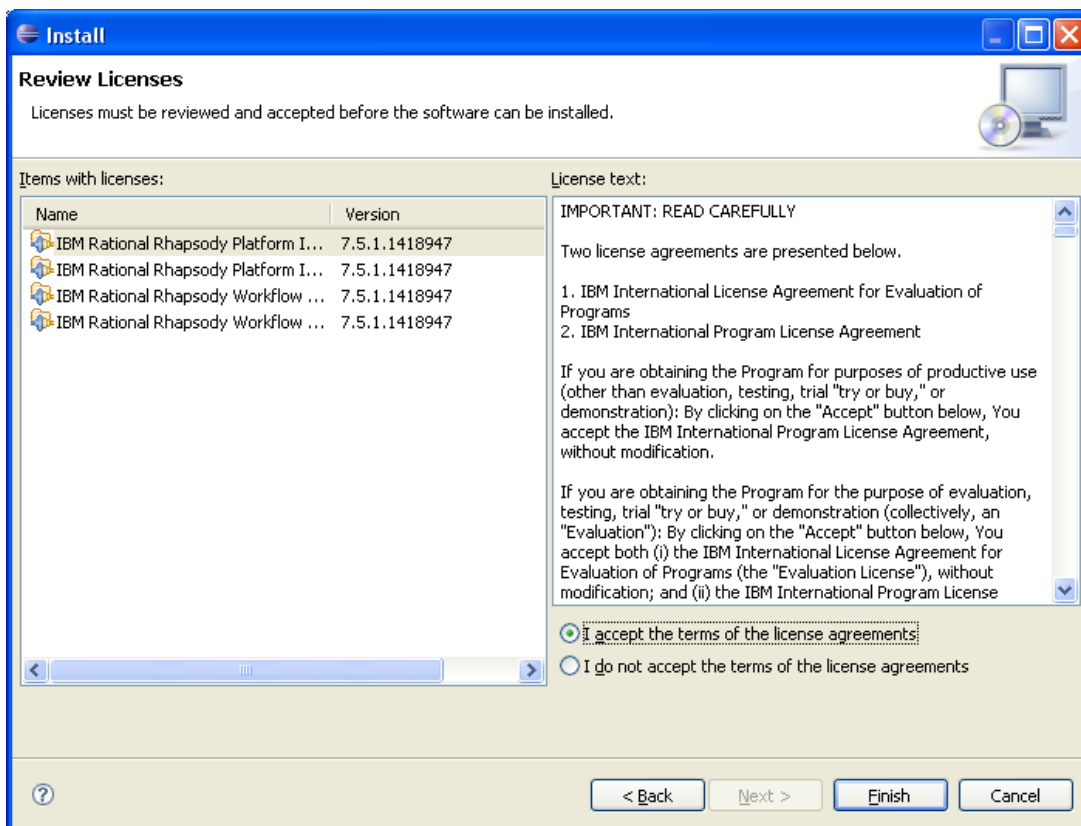
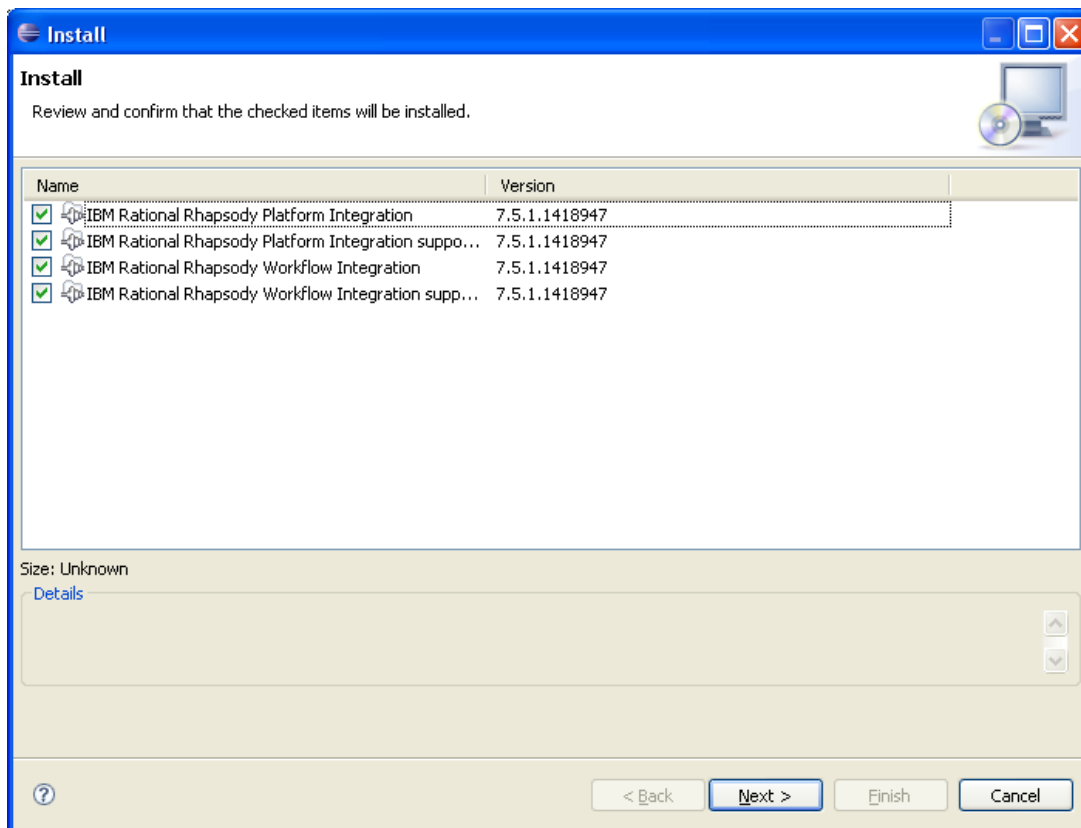


III.4 install Rhp

In Eclipse open the install wizard, click on “add site...”, and select the path of eclipse folder in Rhp install folder



Select IBM Rational Rhapsody Platform Integration
Press install.



IV Use RiA

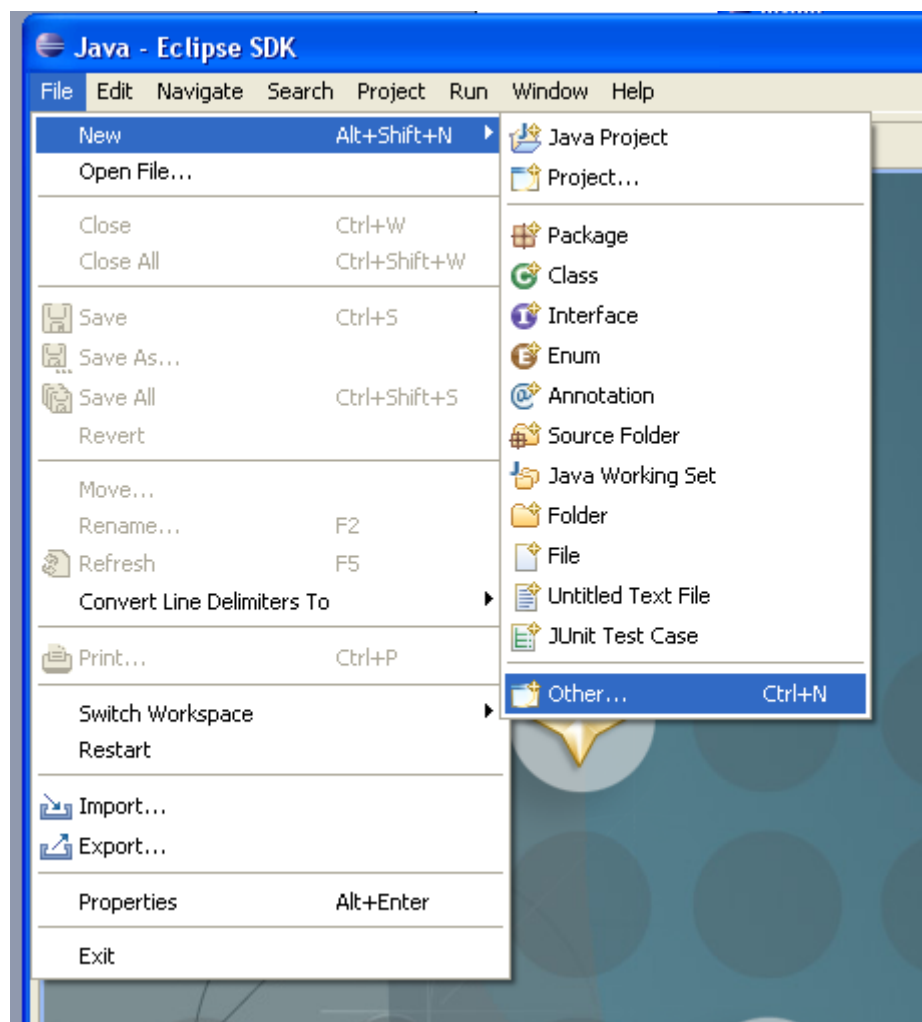
IV.1 Create an Ada project

Now create a new Rhp project

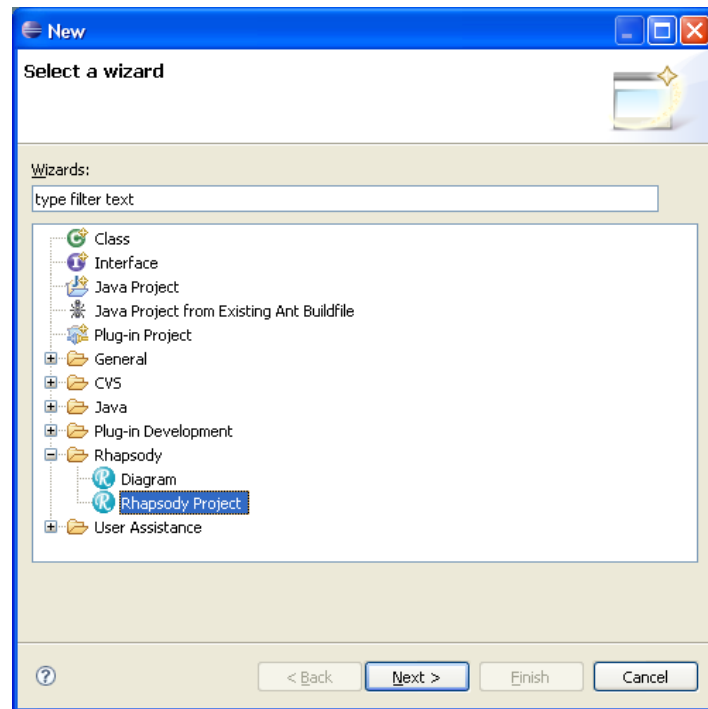
Caution

To create an Ada model, Rhapsody.ini file must be updated with default language equals to ada, before opening Eclipse.

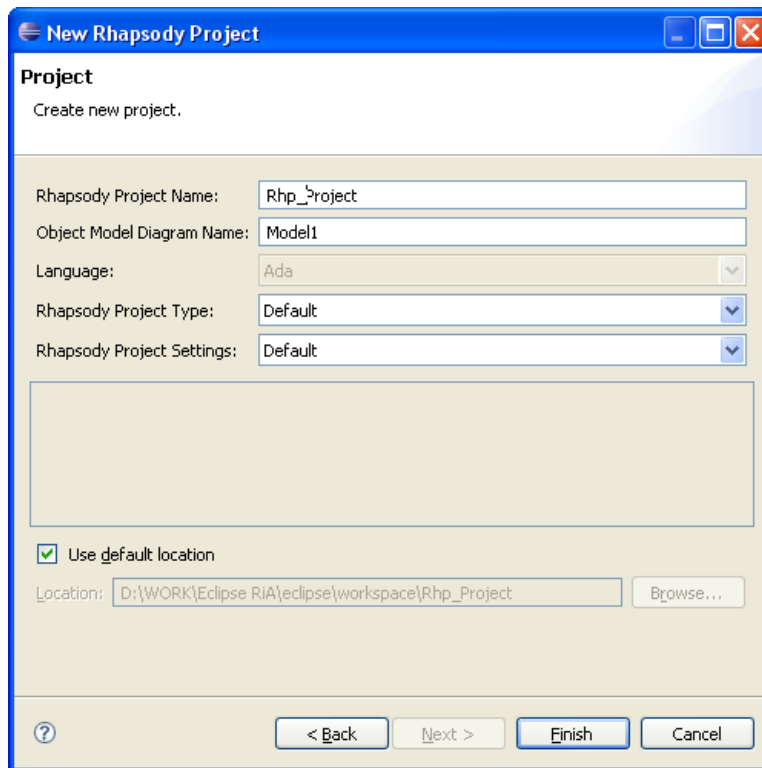
Select menu File/New/Other...



Choose Rhapsody Project



Set the name of your project and press “Finish”

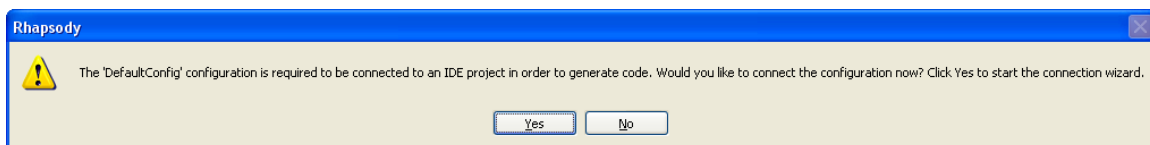


You can now create your Rhp model as usual.

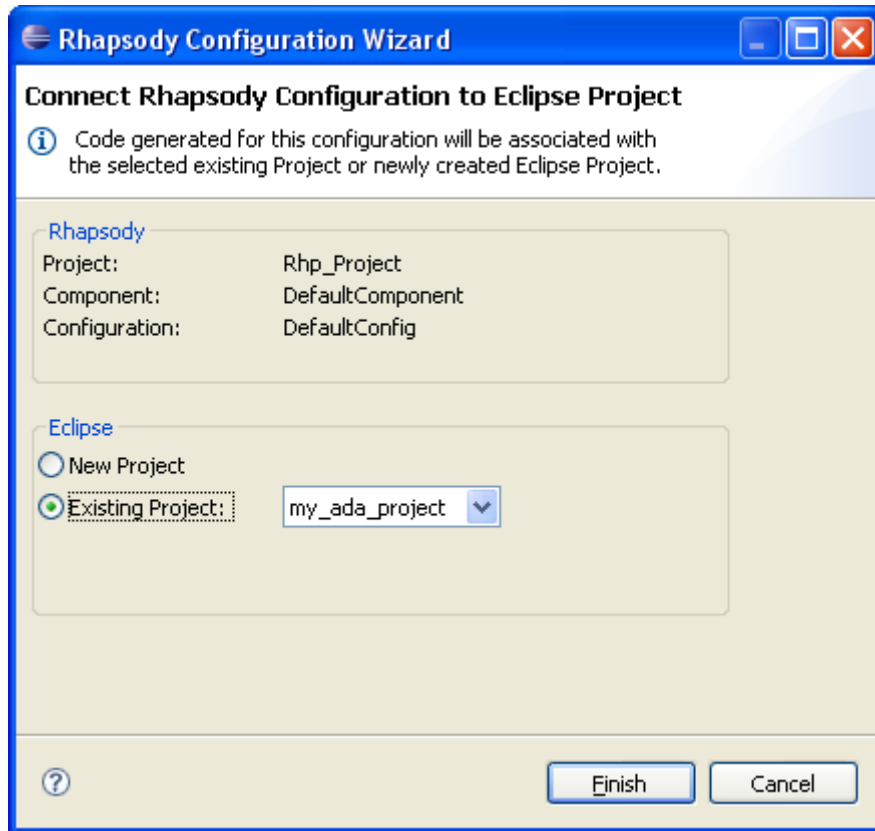
IV.2 Create Ada project

When you will need to generate the code, select menu
code generator/regenerate/<your config>

A popup window will ask you to connect the configuration to an IDE project. Click yes



Select the Ada project which already exists or create a new one.



Ada project contains 2 Files which must be updated by code generator

- <Ada_project_name>.gpr
- Makefile

<Ada_project_name>.gpr

This file describes project configuration. In order to build the Rhp project, the Ada project must be updated with generated folders

<Ada_project_name>.gpr

```
project My_Ada_Project is
    for Main use ("MainDefaultComponent.adb");
    for Object_Dir use "objs";
    for Source_Dirs use ("D:/WORK/Eclipse
RiA/eclipse/workspace/my_ada_project", "D:/WORK/Eclipse
RiA/eclipse/workspace/my_ada_project/default");
```

```

package Compiler is
  for Default_Switches ("ada") use ("-g", "-gnato", "-gnatwa", "-fstack-check");
end Compiler;

package Builder is
  for Default_Switches ("ada") use ("-g", "-gnatQ");
end Builder;

package Ide is
  for Compiler_Command ("ada") use "gnatmake";
end Ide;

end My_Ada_Project;

```

The <Ada_project_name>.gpr file is updated with generated folders.

Makefile

The Ada project make file can be also updated with the Rhp command .
The build: label of this file calls GNAT builder. It can be replaced by our usual build command.

```

# Build executables for all mains defined by the project.
build:
#      $(GNATMAKE) -P "$(GPRPATH) "
#      "C:\Program
Files\IBM\Rational\Rhapsody\7.5.1\share\etc\Executer.exe" "\"C:\Program
Files\IBM\Rational\Rhapsody\7.5.1\share\etc\GnatMake.bat\"
DefaultComponent.bat

```

In order to update those 2 files, code generator can be updated with custom makefiles
We should add new properties to generate Makefile and <Ada_project_name>.gpr files.

The new properties should be added in <Rhp_install_dir>/Share/properties/siteAda.prp file. Here is an example of the definition of new properties

```

Subject Ada_CG
  Metaclass Configuration
    Property Environment Enum
      "GNAT, INTEGRITY, INTEGRITY5, MultiWin32, Multi4Win32, OBJECTADA, RAVEN_PPC, SPARK, GNATVxWorks, Test_Env, GNAT_Eclipse" "GNAT"
    end

```

```

Metaclasse GNAT_Eclipse
  Property AdditionalReservedWords String ""
  Property AdaPathContent MultiLine "<\"\\n\"><\",><\"\\.\"><\"
$AdaCGIDEWorkspace/$AdaCGIDEProject\"><\"\\\\\"><\"/\"><\",><\"\\\\\",
\\\\\">\"
    Property BuildCommandSet Enum "Debug,Release" "Debug"
    Property CompileCommand MultiLine ""
    Property CompileSwitches MultiLine ""
    Property CPPCompileDebug String ""
    Property CPPCompileRelease String ""
    Property DependencyRule String ""
    Property EntryPoint String ""
    Property ErrorMessageTokensFormat String
"TotalNumberOfTokens=2,FileTokenPosition=1,LineTokenPosition=2"
    Property ExeExtension String ".exe"
    Property FileDependencies String ""
    Property ImpExtension String ".adb"
    Property Include String ""
    Property InvokeExecutable String "\"$executable\""
    Property InvokeMake String "\"$OMROOT/etc/Executer.exe\"
\\\"$OMROOT\\etc\\GnatMake.bat\\\" $makefile $maketarget\"
    Property IsFileNameShort Bool "False"
    Property LibExtension String ".a"
    Property LinkDebug String ""
    Property LinkRelease String ""
    Property LinkSwitches String ""
    Property MakeExtension String ".bat"
    Property MakeFileContent MultiLine ""
    Property MakeFileNameForExe1 String "gnat.adc"
    Property MakeFileContentForExe1 MultiLine "$AdaCGGnatAdc"
    Property MakeFileNameForExe2 String "$ComponentName$MakeExtension"
    Property MakeFileContentForExe2 MultiLine "$AdaCGGnatMakefile"
    Property MakeFileNameForExe3 String "Makefile"
    Property MakeFileContentForExe3 MultiLine "# You may edit this makefile
as long as you keep these original
# target names defined.

# Not intended for manual invocation.
# Invoked if automatic builds are enabled.
# Analyzes only on those sources that have changed.
# Does not build executables.
autobuild:
    $(GNATMAKE) -gnatc -c -k -P \"$(GPRPATH)\"

# Clean the root project of all build products.
clean:
    $(GNATCLEAN) -P \"$(GPRPATH)\"

# Clean root project and all imported projects too.
clean_tree:
    $(GNATCLEAN) -P \"$(GPRPATH)\" -r

# Check *all* sources for errors, even those not changed.
# Does not build executables.
analyze:
    $(GNATMAKE) -f -gnatc -c -k -P \"$(GPRPATH)\"

```

```

# Build executables for all mains defined by the project.
build:
#      $(GNATMAKE) -P \"$(GPRPATH)\
      $OMROOT\etc\Executer.exe
\"\\\"$AdaCGOMROOTSingleSlashes\etc\GnatMake.bat\\\" $ComponentName$MakeExtension

# Clean, then build executables for all mains defined by the project.
rebuild: clean build

"
    Property MakeFileNameForExe4 String "ada_project.gpr"
    Property MakeFileContentForExe4 MultiLine "project $AdaCGIDEProject is

for Main use (\"Main$ComponentName.adb\");

package Compiler is
    for Default_Switches (\"ada\") use (\"-g\", \"-gnato\", \"-gnatwa\", \"-fstack-
check\");
end Compiler;

package Builder is
    for Default_Switches (\"ada\") use (\"-g\", \"-gnatQ\");
end Builder;

package Ide is
    for Compiler_Command (\"ada\") use \"gnatmake\";
end Ide;

for Source_Dirs use (\"$AdaCGAdaPath\");
end $AdaCGIDEProject;
"
    Property MakeFileNameForLib String "$ComponentName$MakeExtension"
    Property MakeFileContentForLib MultiLine "$AdaCGGnatMakefile"
    Property OSFileSystemCaseSensitive Bool "False"
    Property ObjCleanCommand String ""
    Property ObjExtension String ".o"
    Property ObjectName String ""
    Property ObjectsDirectory String ""
    Property ParseErrorMessage String "([^\:]+)[:]([0-9]+)[:]\""
    Property QuoteOMROOT Bool "True"
    Property RemoteHost String ""
    Property SpecExtension String ".ads"
    Property UseNonZeroStdInputHandle Bool "True"
    Property UseRemoteHost Bool "False"

end
end

```

IDE project name and IDE workspace location can be read from tags on configuration level.

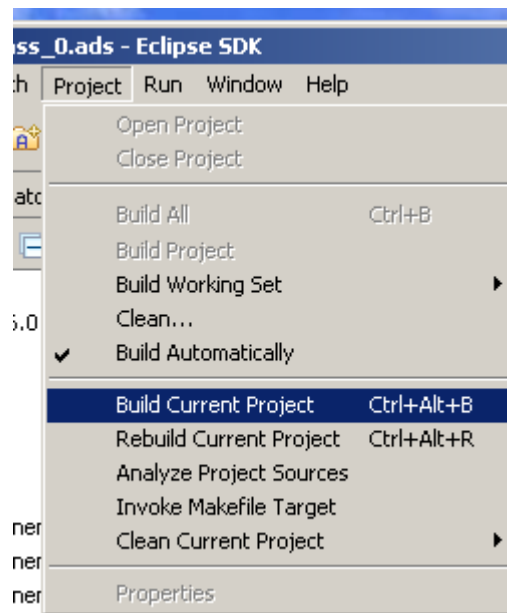
IDEName : Eclipse
IDEProject : Ada project name
IDEWorkspace : workspace path

Some new macros enable user to use them in custom makefiles

AdaCGIDEName
AdaCGIDEProject
AdaCGIDEWorkspace

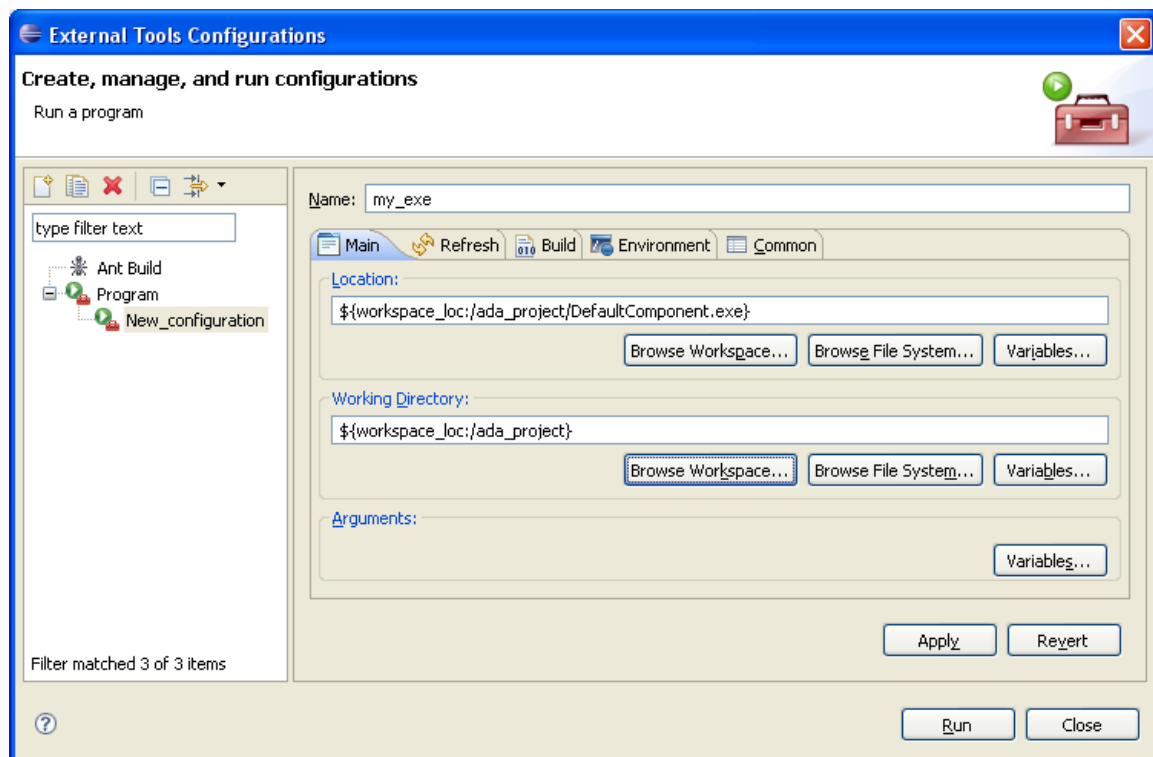
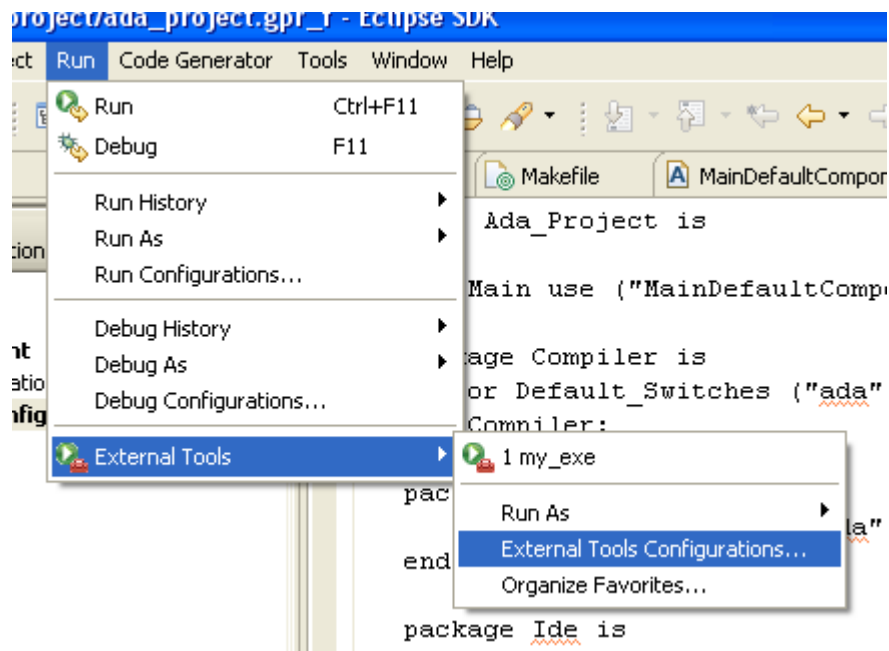
IV.3 Build the project

To build the project, use the menu Project/Build Current Project in an Ada perspective.



IV.4 Run the project

In order to be able to run the model you must configure external tools



V Annexes

Configuring Rational Rhapsody for Eclipse

About this task

To examine the features of an Eclipse configuration in Rational® Rhapsody®:

1. Right-click the Eclipse configuration in the Rational Rhapsody browser.
2. Select Features from the menu to display this window. This version of the Features window contains the **IDE**, **Tags**, **Properties**, and **Settings** tabs for use with Workbench projects.
3. The **IDE** tab provides two important features:
 - **Open in IDE** launches Eclipse with the corresponding project or simply bring the IDE forward.
 - **Build configuration in IDE** sets the build to be performed via the IDE (by sending a request to the IDE)
4. However, if you want to perform the build in Rational Rhapsody, use the **Settings** tab to make the build selections.
5. Generally, you do not need to change the values for the **Tags**, unless you change the *IDEWorkspace directory location*. In that case, you must make the change to the path in the window and click OK.

Rational Rhapsody tags for the Eclipse configuration

After creating an Eclipse configuration, the IBM® Rational® Rhapsody® model elements are coupled with an Eclipse-based project. The definition of the Eclipse project is stored in Rational Rhapsody in these *Tags* (displayed in the browser):

- **IDENAME** is the name of the type of integrated development environment (for example, Eclipse, Workbench).
- **IDEProject** is the project name entered while creating the Eclipse configuration.
- **IDEWorkspace** is the workspace directory for the Eclipse-based project.

The values of the tags are set automatically after the IDE project is created or mapped to the configuration. You do not need to modify the values of these tags,

unless you change the IDE workspace location in the file system. If you make that change, then you must update the IDEWorkspace tag manually. For instructions to make this change if necessary, see [Configuring Rational Rhapsody for Eclipse](#).