Choreographer Platform User Guide

N.V. Haenel, {valentin.haenel@gmx.de}

The Laboratory for Foundations of
Computer Science
School of Informatics
University of Edinburgh

First Edition: January 11, 2005
Revised: Mar 16 , 2005

Contents

1 Introduction

This document describes the Choreographer Platform from the point of view of the user, and is intended to provide assistance for both novice and experienced users of the tool.

The Choreographer is a unified tool for the *DEGAS* (Design Environments for Global ApplicationS) Project [1], combining the tools developed during the Project at the University of Edinburgh, the University of Pisa, and the Technical University of Denmark, into a homogeneous environment

The main purpose of Choreographer is to provide a bridge between the Unified Modelling Language (UML) and process algebras via a set of *Extractors* and *Reflectors*. This means a UML model is annotated with performance security or mobility data in one or more of the diagrams. The extractors are now able to extract a process algebra model from the UML. The extracted model may be analysed for performance, security, and/or mobility results. These results are then reflected to the UML level by the reflectors, to make them visible in the UML diagrams that were initially annotated, and thereby presenting the results of the analysis in a language understood by the UML modeler.

Currently three process algebras have been incorporated, PEPA for performance models [2], LySa for security protocols [3], PEPA Nets for mobility [4]. Respectively three external tools have been integrated into Choreographer, the PEPA Workbench Java Edition, the Lysatool, and the PEPA Workbench for PEPA Nets.

Choreographer is built on the NetBeans Platform [5] A flexible infrastructure for building complex desktop applications.

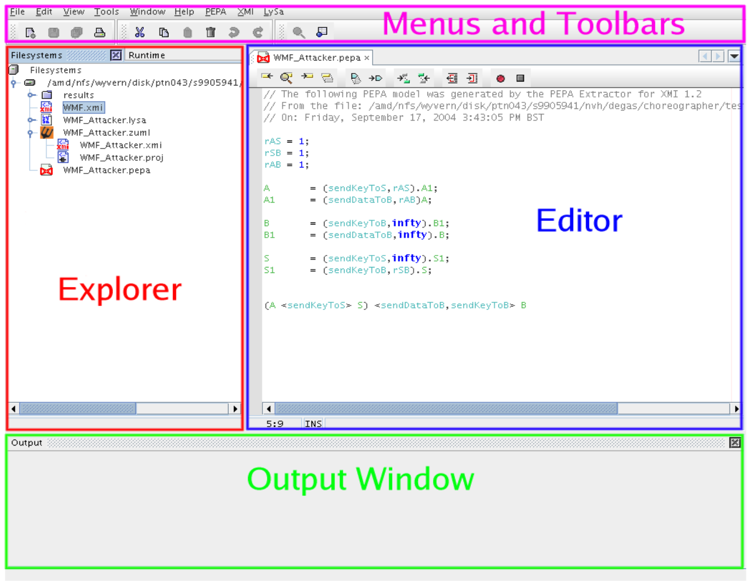
2 Introduction to the NetBeans Platform

This section gives an introduction to the features of the NetBeans Platform.

Figure 1 shows a common screen layout for the Choreographer. Observe the Menus and Tool-bars at the top of the screen. The left hand side is occupied by the Explorer. The right hand side contains the Editor, and the bottom of the screen encompasses the output tab.

NetBeans 3.6 requires a directory to be mounted in order to view or use the files in it. To mount a local directory, right click the icon labelled `Filesystems` at the top of the explorer window. Select the option `mount`, and then `local directory`. Now select the desired directory using the file browser. Once mounted the directory is displayed as a child node of `Filesystems` and can be explored from there.

The Explorer is used to browse any mounted file systems, or local directories, and allows you to inspect their contents. Nodes in the Explorer can be expanded and collapsed to vary the depth of the displayed file system. The Explorer recognises file types and will display an appropriate icon for each different file type. Furthermore all file types have an associated context menu that can be used to invoke actions specific to this file. Figure 2 shows the features of the



```
// The following PEPA model was generated by the PEPA Extractor for XMI 1.2
// From the file: /amd/nfs/wyvern/disk/ptn043/s9905941/nvh/degas/choreographer/tes
// On: Friday, September 17, 2004 3:43:05 PM BST

rAS = 1;
rSB = 1;
rAB = 1;

A       = (sendKeyToS,rAS).A1;
A1      = (sendDataToB,rAB)A;

B       = (sendKeyToB,infty).B1;
B1      = (sendDataToB,infty).B;

S       = (sendKeyToS,infty).S1;
S1      = (sendKeyToB,rSB).S;


(A <sendKeyToS> S) <sendDataToB,sendKeyToB> B
```

Figure 1: The Screen Layout of Choreographer

Explorer in detail.

The Menus and Tool-bars in Choreographer are used to invoke basic file and text operations such as `New File`, `Load File`, `Save File`, `Cut`, `Copy`, `Paste`. `Undo` `Redo` and `Find`. Tool-bars can be enabled and disabled. Figure 3 shows some of the more commonly used tool-bar items. All tool-bar items are equipped with tool-tips that appear when you hover the mouse over the item for a couple of seconds.

The NetBeans Editor is a sophisticated text editor that can be used to view and modify the content of files. Figure 4 shows the most important features of the Editor. Clearly visible, are one active and one inactive file in a tab, the Editor's own tool-bar, the main text window that is currently displaying a PEPA file, and lastly the field in the lower left hand corner that displays the line and column number. The tool-bar contains some useful text editing commands, such as `Find Selection` and `Find Next Occurrence` that allow you to highlight a string within a document and then search the document for this string. All the tool-bar items have tool-tips that appear when you hover the mouse over the item for two seconds.

The Editor itself can be customised in a variety of ways to make your experience more pleasant and productive. Figure 5 shows the multitude of configurable options and settings for the plain text editor. All the entries have associated tool-tips that are displayed at the bottom of the window. The properties window

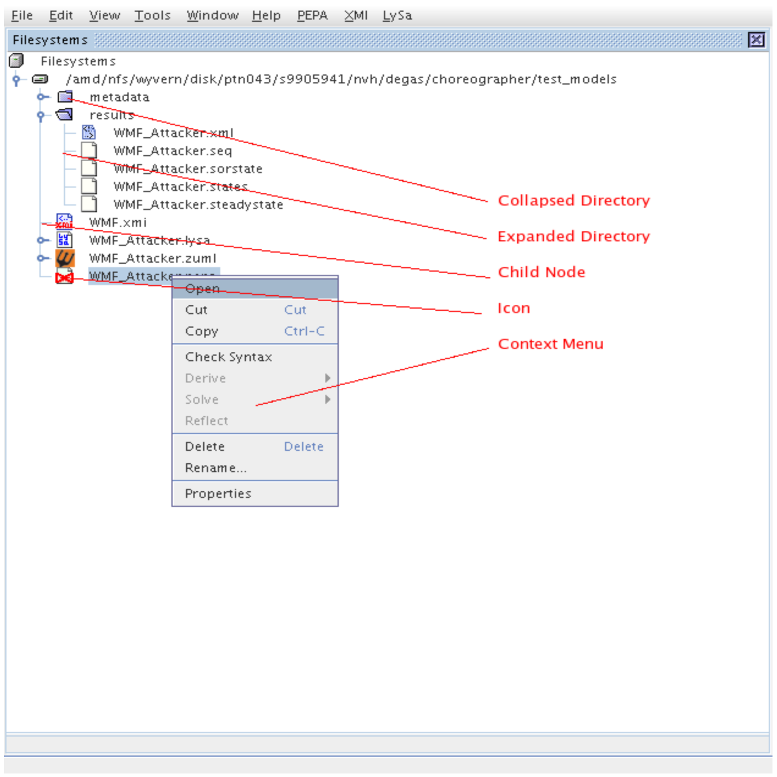


Figure 2: Details of the Explorer

can be found under `Tools --> Options --> Editing --> Editor Options --> Plain editor`. For example the property `Line Number Margin` can be set to `true` to display the line numbers of the document in the left margin of the editor. `Tab Size` denotes how many spaces there are to a single tab by default.

3 Installation Instructions

Choreographer itself has been developed in the Java programming language. The PEPA Workbench is written in Java too. The LySa tool and the PEPA Workbench for PEPA Nets are written in Standard ML(SML). Choreographer will happily work under both Windows and Linux. We used Red Hat Linux, kernel 2.4 and Windows XP(SP1) for development and testing. Choreographer requires:

- An installation of Poseidon for UML, to draw the UML diagrams.[6]

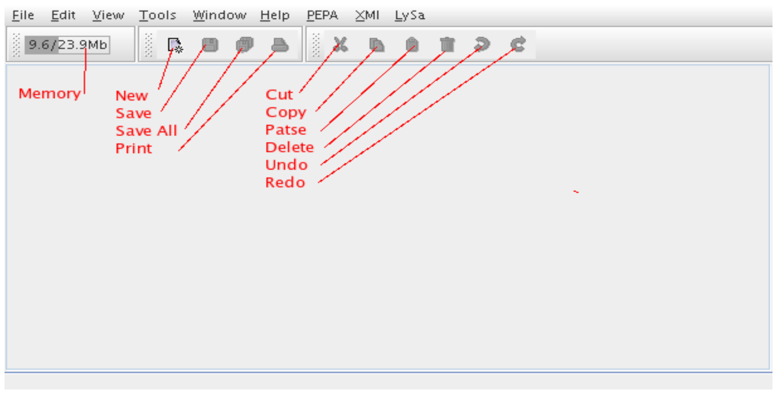
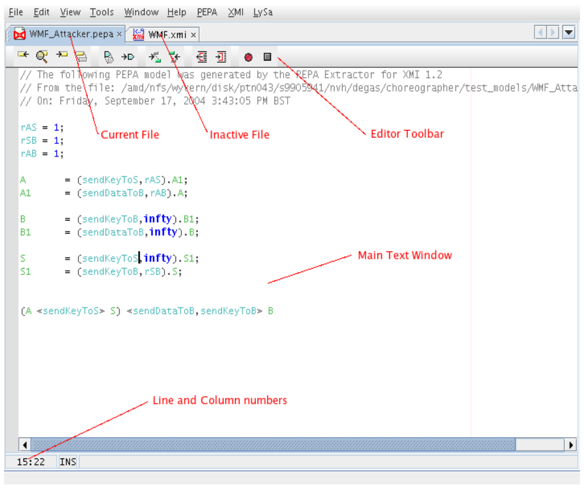


Figure 3: Menus and Toolbars

- An installation of Java 1.4.2 Runtime Environment for the main application. [7]
- An installation of Standard ML of New Jersey, for the LySa and PEPA Net subcomponents. [8]

Once all of the above have been installed, Choreographer itself can be installed.

- Download the latest .zip from the downloads section of the choreographer website at [9].
- Unzip the downloaded file, this will create a directory *choreographer* that contains the application files.
- Launching:
  - For Windows, open the directory *choreographer* in a Windows Explorer and change to the *bin* directory. From there double-click *runidew.exe*.
  - For Linux, open a terminal and cd to the *choreographer/bin* directory, and from there execute *./runide.sh*.
- Set the location of the SML runtime. Select `Tools --> Options --> LySa settings`, and in the properties window, select *SML Runtime*. To set the location click on the white box on the far right hand-side and select the appropriate file in the file browser window. (For Linux this is the file */bin/sml.bin* in the SML installation directory and for Windows it is the file *\bin\sml.bat* in your SML installation directory.)
- This completes the installation and you may now use the tool.

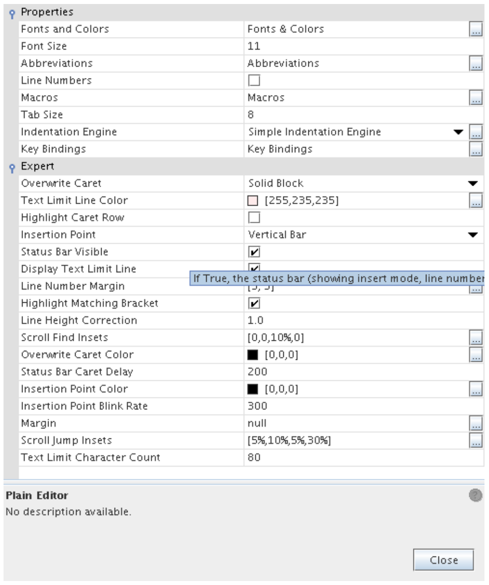


```
// The following PEPA model was generated by the PEPA Extractor for XMI 1.2
// From the file: /amd/nfs/wyvern/disk/ptn043/s9905941/nvh/degas/choreographer/test_models/WMF_Atta
// On: Friday, September 17, 2004 3:43:05 PM BST

rAS = 1;
rSB = 1;
rAB = 1;

A       = (sendKeyToS,rAS).A1;
A1      = (sendDataToB,rAB).A;

B       = (sendKeyToB,infty).B1;
B1      = (sendDataToB,infty).B;

S       = (sendKeyToS,infty).S1;
S1      = (sendKeyToB,rSB).S;


(A <sendKeyToS> S) <sendDataToB,sendKeyToB> B
```

Figure 4: Editor Detail

4 Quick Start

- Download the file *choreographer-models.zip* from the Chreographer website at [9] and unzip it.
- Select the *filesystems* icon in the explorer window on the left hand side of the screen. Click select `mount --> local directory`. Then select the directory that contains the examples and expand it by clicking on the little grey circle in front of the icon. This will display the content of the directory.
- Expand the directory *LySa/WMF/* to display the models suitable for security analysis.
- Now select the file *WMF_00.zuml* and right click on it. The select `Extract LySa` to invoke the extractor. This produces the files *WMF_00.lysa*, and you should see progress messages in the console at the bottom of the screen. Once extracted the resulting model will be opened in the Editor.
- To invoke the LySa tool, select the file *WMF_00.lysa*, and right click. Then select `Invoke LySa` from the menu to run the tool. Any output appears in the console. For this example the tool will determine that the protocol has possible errors. An example with errors has been chosen, as LySa models without errors provide no results that could be reflected.
- Reflection is now possible, right click on *WMF_00.lysa* and select `Reflect`. You will now be presented with two file browsers, the first allows you



Properties	
Fonts and Colors	Fonts & Colors
Font Size	11
Abbreviations	Abbreviations
Line Numbers	☐
Macros	Macros
Tab Size	8
Indentation Engine	Simple Indentation Engine
Key Bindings	Key Bindings
Expert	
Overwrite Caret	Solid Block
Text Limit Line Color	☐ [255,235,235]
Highlight Caret Row	☐
Insertion Point	Vertical Bar
Status Bar Visible	✓
Display Text Limit Line	✓
Line Number Margin	[5, 5]
Highlight Matching Bracket	✓
Line Height Correction	1.0
Scroll Find Insets	[0,0,10%,0]
Overwrite Caret Color	■ [0,0,0]
Status Bar Caret Delay	200
Insertion Point Color	■ [0,0,0]
Insertion Point Blink Rate	300
Margin	null
Scroll Jump Insets	[5%,10%,5%,30%]
Text Limit Character Count	80

If True, the status bar (showing insert mode, line number

Plain Editor

No description available.

Figure 5: Editor Properties

to select the original file that contains the model, and the second allows you to enter a file name for the result of the reflection procedure. In the first file browser select the file *WMF_00.zuml*, and in the second one leave the default as *WMF_00.zuml*. This will produce the file *WMF_00.security.zuml*, that contains the results from the analysis. The original file *WMF_Attacker.zuml* remains intact.

- Now expand the directory *PEPA*.
- Now select the file *WMF_Attacker.zuml* and right click on it. The select `Extract PEPA` to invoke the extractor. This produces the files *WMF_Attacker.pepa*, and you should see progress messages in the console at the bottom of the screen. Once extracted the resulting model will be opened in the Editor.
- Now select the file *WMF_Attacker.pepa* and right click to show the menu. To solve the model:
  - `Select Check Syntax`
  - `Select Derive --> Silent`
  - `Select Solve --> SOR`

  Progress messages are displayed in the console as the tool advances.
- Reflection is now possible: right click on *WMF_Attacker.pepa*, and select `Reflect`. Again you will be presented with two file browsers, the first for

selecting the original, and the second for selecting the destination file for the reflection. For the first, select *WMF_Attacker.zuml*, and then leave the default name as *WMF_Attacker.performance.zuml*. This produces the file *WMF_Attacker.performance.zuml* that contains the results from the PEPA reflection.

- Expand the directory *PEPA-Nets*.
- Now select the file *JourneyOfLetter1-Version1.zuml* and right click on it. The select `Extract PEPA Net` to invoke the extractor. This produces the files *JourneyOfLetter1-Version.zuml*, and you should see progress messages in the console at the bottom of the screen. Once extracted the resulting model will be opened in the Editor.
- Now select the file *JourneyOfLetter1-Version.pepanet* and right click to show the menu. To solve the model:
  - `Select Generate XML Output`
  - `Select Load Matrix`
  - `Select Solve Matrix`

  Progress messages are displayed in the console as the tool advances.
- Reflection is now possible: right click on *JourneyOfLetter1-Version.pepanet*, and select `Reflect`. Again you will be presented with two file browsers, the first for selecting the original, and the second for selecting the destination file for the reflection. For the first, select *JourneyOfLetter1-Version.zuml*, and then leave the default name as *JourneyOfLetter1-Version.mobility.zuml*. This produces the file *JourneyOfLetter1-Version.mobility.zuml* that contains the results from the PEPA Net reflection.
- Lastly view the reflected models in Poseidon to convince yourself that it has worked.

5 Support for the PEPA formalism

This section describes the functionality offered by the PEPA module within Choreographer

The PEPA module of Choreographer is an interface layer that binds the *PEPA Workbench Java Edition* [10] into the Choreographer framework. Support for the PEPA formalism is provided partly by the PEPA Workbench, and partly by the Choreographer platform.

The PEPA menu has the following layout:

- Check Syntax
- Derive

  - Silent
  - Compressed
  - Uncompressed
- Solve
  - LNBCG Solve
  - SOR Solve
- PEPA State Finder
- Throughput
- Single Step Debugger
- Reflect

These menu items are displayed in two distinct places, the PEPA menu in the menu-bar at the top of the application, and in the context menu for PEPA files in the Explorer. PEPA files are displayed using a butterfly combinator icon in the Explorer. Throughout the next section assume that `example.pepa` is the model being used.

The Choreographer provides some simple syntax checking that can be performed to assert the lexical and the syntactical correctness of the model. To invoke this from the menu select `Check Syntax`, the results of the check will be displayed in the output window.

The Choreographer provides State Space Derivation in three output formats, `Silent`, `Compressed` and `Uncompressed`. This will derive the state space of the model. `Silent` produces no output, only an internal representation of the state space. `Compressed` will produce a compressed version of the state space in the file *results/example.states*. This lists all the states but the sequential derivatives have been replaced with integer encodings. In addition a file *results/example.seq* is produced. This contains the integer encodings of the sequential derivatives and allows the compressed state space to be decoded. `Uncompressed` will produce the file *results/example.states*: this time the state space has not been encoded, and the file lists all possible states of the PEPA model in verbose detail. In all cases progress messages will be displayed in the output window.

There are two available solvers for steady state solution, the linear biconjugate gradient method, and the successive over-relaxation method. The first is known to converge faster and use less memory and is therefore recommended. The steady state solution will be written to a file *results/example.steadystate* for the linear biconjugate gradient method, and *results/example.sorstate* for the successive over-relaxation. . To invoke the solvers, select either `Solve --> LNBCG Solve` or `Solve --> SOR Solve`. Progress messages will be displayed in the output window. The settings for the two solvers can be accessed through `Tools --> Options --> PEPA Module Settings --> (LNBCG/SOR) settings`.

`Reflect` is related to the Extractor/Reflector support offered by Choreographer and is described in Section 8.

Verbose mode for the PEPA module can be activated through `Tools --> Options --> PEPA Module Settings --> General --> Verbose`. This enables verbose output for the PEPA extractor, PEPA reflector, and the PEPA Workbench.

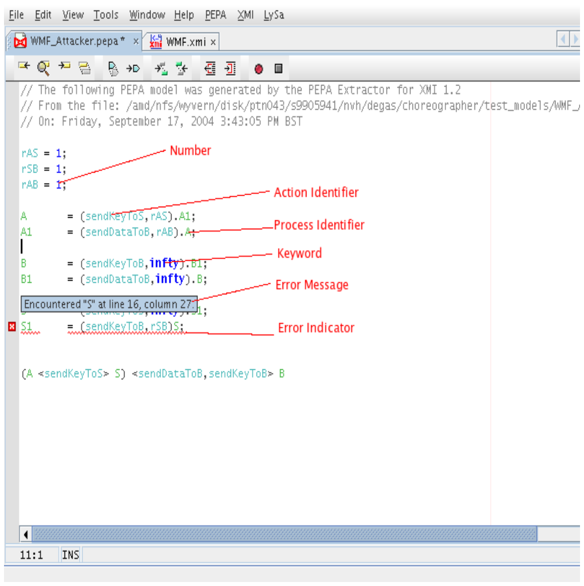
The Choreographer provides syntax highlighting for the PEPA formalism. This means that certain classes of identifiers, such as component and action identifiers, the PEPA keywords such as `infty`, operators such as `+`, all numerical values and comments, are displayed in differing colours. This makes it easier and more comfortable to work with PEPA models. Syntax highlighting is enabled by default, and cannot be disabled. Figure 6 shows the syntax highlighting. The syntax highlighting settings for the editor may be modified. Select `Tools --> Options --> Editing --> Editor Settings --> PEPA Editor.` This will provide you with a settings panel to tweak the settings for the PEPA editor. To change the default highlighting, select `Fonts and Colours`. This will open up a list of categories, such as `pepa-comment` or `pepa-keyword`. Each of these categories has a default font, size and colour that can be customised. Figure 7 shows the fonts and colours window for PEPA.

The Choreographer provides background parsing for PEPA files while they are being edited. This means that Choreographer will perform a syntax check at regular intervals while the file is being modified in the editor. To be precise: every time a single character modification to the file is made, a timer is restarted, when the timer expires, the automatic syntax check kicks in. If any syntax mistakes are found the line in question is underlined clearly with a red wave like graphic, and a red cross appears in the margin of the editor. If you now hover the mouse over the red cross, a short description of the syntax error is displayed. Figure 6 shows a syntax mistake, the underline graphic and the short error message. The settings for the background parsing can be customised using `Tools --> Options --> PEPA Module Settings --> General`. This allows you to enable/disable background parsing and set the delay for the timer.

More information about the state space derivation and the two solvers is included in the User Manual for the PEPA workbench that can be obtained from [10].

5.1 Advanced Dialogues

The three items `PEPA State Finder`, `Throughput` and `Single Step Debugger` are dialogues for advanced PEPA analysis. These are dialogue windows of the PEPA Workbench, that have been incorporated into the Choreographer Platform. They provide the exact same functionality as in the PEPA Workbench. The dialogues simply open as components within Choreographer, and can be moved around and repositioned within the main window as shown in Figure 8



```
// The following PEPA model was generated by the PEPA Extractor for XMI 1.2
// From the file: /amd/nfs/wyvern/disk/ptn043/s9905941/nvh/degas/choreographer/test_models/WMF_
// On: Friday, September 17, 2004 3:43:05 PM BST

rAS = 1;
rSB = 1;
rAB = 1;

A       = (sendKeyToS,rAS).A1;
A1      = (sendDataToB,rAB).A;

B       = (sendKeyToB,infty).B1;
B1      = (sendDataToB,infty).B;

Encountered "S" at line 16, column 27:
S1      = (sendKeyToB,rSB)S;


(A <sendKeyToS> S) <sendDataToB,sendKeyToB> B
```

Figure 6: PEPA Editor Details

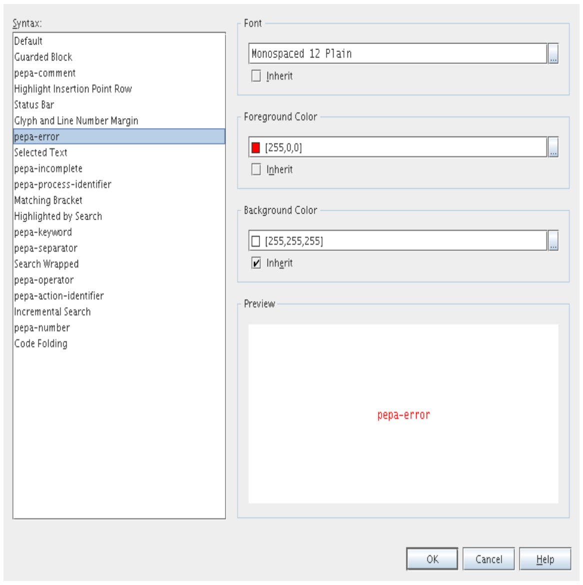
5.2 The PEPA State Finder

This component allows you to find the steady state probability associated with either a single state, or a set of states specified by a pattern.

The state finder consists of one text field at the top that is labelled *Sequential Representation*. This contains a schematic representation of the system equation, where all the sequential components are replaced by the string "SEQ". Underneath the Sequential Representation there are two multiline text panels. The first contains the derivatives of all sequential components within the model. The second is an input panel for the state or the pattern of states that the state finder should match. To-wards the bottom you will find another single line text field to display the resulting total probability and three buttons: `Run`, `Clear` and `Quit` to control the operation.

The state finder requires solving of the model within the workbench, as it simply looks for states matching the pattern, and then sums their probabilities. If the steady state solution has not yet been computed, the state finder will discover all states matching the pattern only, without a numerical answer. The matched states and their total probability will be written to the file *results/example.psf*. Importantly the PEPA State finder requires the model to be successfully solved for steady state, so that it can find the matching probabilities. The numerical answer for total probability is also displayed in the state finder's window if it exists.

To initiate a pattern, single click on one of the sequential derivatives to add



Font: Monospaced 12 Plain
☐ Inherit

Foreground Color: 🟥 [255,0,0]
☐ Inherit

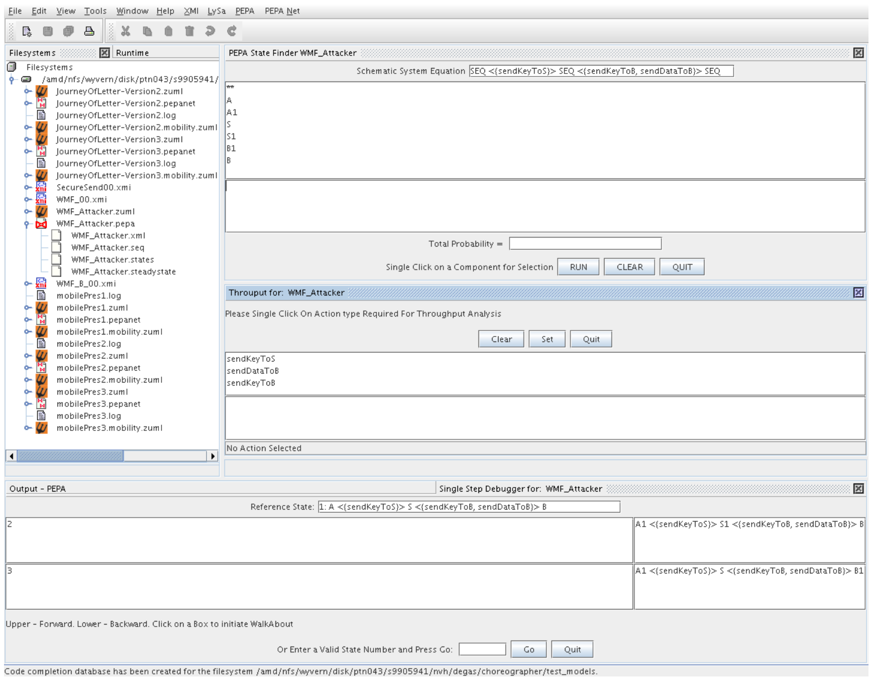
Background Color: ☐ [255,255,255]
☑ Inherit

Figure 7: PEPA Editor Fonts and Colours Detail

this derivative to the pattern in the lower window. The string "**" denotes a wild card and will match any of the sequential derivatives. Any cooperation operators in the system equation will automatically be inserted for you when they are required. Simply add either concrete derivatives or the wild card to the pattern until you are happy with it. Then, click `Run` to invoke the state finder. If you would like to re-run the state finder, click `Clear` and enter a new state or pattern.

5.3 Single Step Debugger

The Single Step Debugger can be used to traverse the state space of the model, one state at a time. The debugger will display the current state at the top of the screen. The upper two text panels display the states that can be stepped into for-wards, the state number on the left and the state representation on the right. The lower two panels display the states that can be stepped into back-wards, again state number on the left and state representation on the right. At the bottom there is an input field that allows you to jump directly to a state with a given number. To jump to a state, enter that state's number and click `Go`. To walk about the state space, single click on any state in the upper or lower panels to move to that state. The single step debugger is useful for models that are deadlocked as it allows you to examine closely where the deadlock occurs, and how the deadlocked state was reached.


File Edit View Tools Window Help XMI LySa PEPA PEPA Net
Filesystems
Runtime
/amd/nfs/wyvern/disk/ptn043/s9905941/
JourneyOfLetter-Version2.zuml
JourneyOfLetter-Version2.pepanet
JourneyOfLetter-Version2.log
JourneyOfLetter-Version2.mobility.zuml
JourneyOfLetter-Version3.zuml
JourneyOfLetter-Version3.pepanet
JourneyOfLetter-Version3.log
JourneyOfLetter-Version3.mobility.zuml
SecureSend00.xmi
WMF_00.xmi
WMF_Attacker.zuml
WMF_Attacker.pepa
WMF_Attacker.xml
WMF_Attacker.seq
WMF_Attacker.states
WMF_Attacker.steadystate
WMF_B_00.xmi
mobilePres1.log
mobilePres1.zuml
mobilePres1.pepanet
mobilePres1.mobility.zuml
mobilePres2.log
mobilePres2.zuml
mobilePres2.pepanet
mobilePres2.mobility.zuml
mobilePres3.zuml
mobilePres3.pepanet
mobilePres3.log
mobilePres3.mobility.zuml
RUN
CLEAR
QUIT
Clear
Set
Quit
Output - PEPA
Go
Quit

PEPA State Finder WMF_Attacker

Schematic System Equation: SEQ <{sendKeyToS}> SEQ <{sendKeyToB, sendDataToB}> SEQ

```
**
A
A1
S
S1
B1
B
```

Total Probability =

Single Click on a Component for Selection

Throuput for: WMF_Attacker

Please Single Click On Action type Required For Throughput Analysis

```
sendKeyToS
sendDataToB
sendKeyToB
```

No Action Selected

Single Step Debugger for: WMF_Attacker

Reference State: 1: A <{sendKeyToS}> S <{sendKeyToB, sendDataToB}> B

2	A1 <{sendKeyToS}> S1 <{sendKeyToB, sendDataToB}> B
3	A1 <{sendKeyToS}> S <{sendKeyToB, sendDataToB}> B1

Upper - Forward. Lower - Backward. Click on a Box to initiate WalkAbout

Or Enter a Valid State Number and Press Go:

Code completion database has been created for the filesystem /amd/nfs/wyvern/disk/ptn043/s9905941/nvh/degas/choreographer/test_models.

Figure 8: PEPA Advanced Dialogues

5.4 Throughput Selection

This component allows selection of action types for throughput analysis.

This dialogue contains two list-boxes. The upper one contains action types that exist in the model but not will consider in calculation. The lower contains selected action types that will calculated. Action types can be selected/deselected with a single click. The button `Clear` disposes all action types. The button `Quit` save selection and close dialogue. If you open this dialogue in future all previously selected action types will saved.

Calculation of throughput parameters is carried out simultaneously with deriving and solving. If the model file is called *model.pepa*, The result file is called *model.throughput* and contains both flow of probability per action type per state and total throughput of each action type.

For each action type flow of probability per action, per state is calculated using the steady state output. Total throughput for an action type is the sum of the flow of probabilities for each action type, for each state.

6 Support for the LySa formalism

This section describes the support offered by the LySa module within Choreographer.

The LySa module offers support for the LySa formalism. This is achieved by binding the *Lysatool* [11] into the Choreographer framework.

The LySa menu layout is as follows:

- Invoke LySa Tool
- Reflect

This menu is accessible from the LySa menu item in the menu-bar, and from the context menu for LySa files in the Explorer. Lysa files are recognised by the Explorer and displayed with the LySa icon.

The command `Invoke LySa` will run the LySa tool on the selected file. The support for PEPA allows you to complete the model analysis step-by-step, i.e. check syntax, derive state space and solve model, and therefore provides a more elaborate menu. The LySa support performs the equivalent steps for the LySa formalism with a single command. Thus invoking the LySa tool will perform a syntax check and also analyse the model. If the syntax check fails the LySa tool will report a diagnostic error message including the line number. Any messages output by the LySa tool are forwarded to the output tab of Choreographer.

The `Reflect` command is used in conjunction with the Reflector for LySa, and is described in Section 8.

The support for the LySa formalism offers simple syntax highlighting, in much the same way that it is provided for PEPA. The difference is that no background parsing is provided for LySa, and all syntax checking is performed by the tool itself.

The editor and default syntax colours can be customised the same way they are customised for PEPA. However the settings are located in: `Tools --> Options --> Editing --> Editor Settings --> LySa Editor`.

There are a small number of settings to customise the LySa support offered by Choreographer that can be found in: `Tools --> Options --> LySa Settings`. The most important option is the `Trace Output` option, that allows you to enable verbose output for the LySa Extractor. There are two other options, `LySa Heap Image` and `number of instances, n`. However these are undocumented features used in debugging and development.

7 Support for the PEPA Net formalism

This section describes the functionality provided by the PEPA Nets module within Choreographer.

Support for the PEPA Net formalism is provided by the PEPA Net module. This binds the Standard ML application, `The PEPA Workbench for PEPA Nets` into the Choreographer. Currently the tool can be downloaded from the *Tools* section of the PEPA website at [2] and a good overview of its uses can be found in [4]

The PEPA Net menu has the following options:

- Invoke Workbench for PEPA Nets
- Generate XML output
- Load Matrix
- Solve Matrix
  - LNBCG Solve
  - SOR Solve
- Reflect

The basic architecture for the PEPA Net support, including extraction, and the tool chain used is shown in Figure 9.

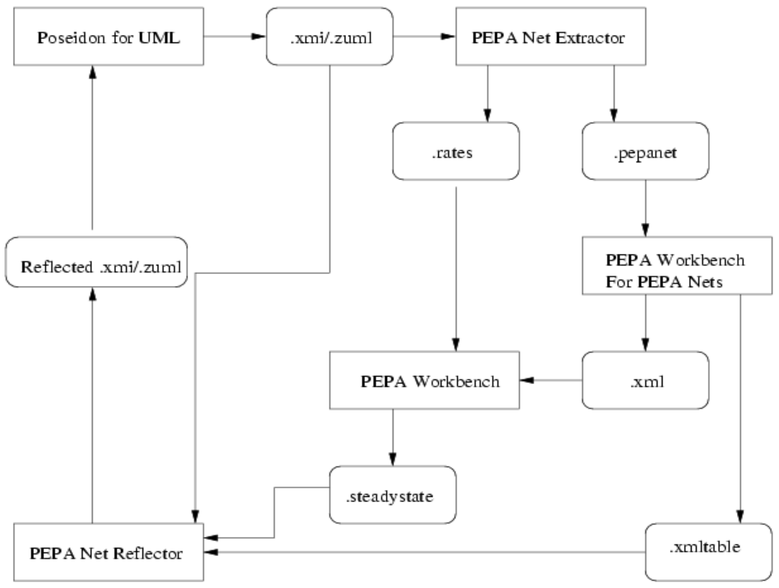


Figure 9: Architecture for PEPA Nets

Currently the only tool that can explore the state space of PEPA Net model is the PEPA Workbench for PEPA Nets. However, this tool is not equipped with numerical solvers, and thus we had to outsource the solvers from the PEPA Workbench Java Edition. To facilitate this process, we modified the PEPA Workbench for PEPA Nets to produce the underlying Markov chain in XML format, and modified the PEPA Workbench to load Markov chains in XML format. Thus we use the PEPA Workbench for PEPA Nets to explore the state space, and the PEPA Workbench to solve for steady state solution.

The complete picture is as follows.

- Use `Poseidon` to generate an *.xmi* or *.zuml* file
- Use the `PEPA Net Extractor` to produce a *.pepanet* and a *.rates* file.
- Use the `PEPA Workbench for PEPA Nets` to explore the state space of the *.pepanet* file and produce the Markov chain as *.xml* file and the state space representation in the file *.xmltable*
- Use the Markov chain *.xml* file, and the *.rates* file to load the Markov chain into the `PEPA Workbench`
- Use the `PEPA Workbench` to solve for steady state, producing a *.steadystate* file
- Use the *.steadystate*, *.xmltable*, and original *.xmi/.zuml* file as input to the `PEPA Net Reflector`, producing a reflected version, that can be loaded into `Poseidon` again.

This corresponds loosely to the PEPA Net menu in Choreographer.

`Generate XML Output` will run the PEPA Workbench for PEPA Nets to produce the Markov chain in XML format. `Load Matrix` will load the Markov chain into the PEPA Workbench. `Solve Matrix` will solve the Markov chain using the solvers of the PEPA Workbench. `Reflect` will invoke the PEPA Net extractor.

There is one additional item, namely `Invoke PEPA Workbench` that can be used to execute the workbench with arbitrary options. To facilitate this a settings tab for the PEPA Net module can be found in `Tools --> Options --> PEPA Net Settings`. The settings window can be used to configure the flags passed to the PEPA Workbench for PEPA Nets if the command `Invoke PEPA Workbench` is executed. The flags allow for a variety of different behaviours of the tool allowing you to really run the PEPA Workbench for PEPA Nets from Choreographer. For example to compile a PEPA Net model to the equivalent PEPA model, open the settings tab and check the item `compile` and then execute the command `Invoke PEPA Workbench` to run the PEPA Workbench for PEPA Nets using the `compile` flag.

The current flags offered are:

```
Usage:
  pwb [options] filename.pepa
where options include
  -help              Print this message and exit
  -version           Print version number information and exit
  -silent            Run silently, produce no console messages
  -nohashing         Do not write a hash table file
  -statespaceonly    Do not write a transition matrix file
  -aggregate         Try to aggregate the model to reduce state space
  -aggregating       Provided as a synonym for -aggregate
  -viewintermediate  View intermediate transformations when aggregating
```

```
-branching         Print branching information for each state
-xmloutput         Write output in XML format
-maple             Write output in Maple(TM) format
-matlab            Write output in Matlab(TM) format
-mathematica       Write output in Mathematica(TM) format
-priorities        Allow use of priorities on firings
-compile           Compile a PEPA net to an equivalent PEPA model
-bridge            Bridge between the PEPA Workbench and Dizzy simulator
-listing           Generate a compiler listing when compiling to PEPA
```

8 Extraction, Reflection and XMI

XMI is an acronym for XML Metadata interchange format, an XML based language for expressing UML models in a standard syntax, and the extension *xmi*. This is the format used by many UML tools for serialising models and saving them. Poseidon for UML uses a simple zip format with the extension *zuml*, containing the XMI specification of the UML model. Users of Choreographer may work with both *.zuml* and *.xmi* formats.

Apart from the process calculus tools, Choreographer also unifies several pairs of *Extractors* and *Reflectors*. These are software tools that provide a bridge between the UML paradigm used for specification and the process calculus paradigm used for verification. Figure 10 shows a conceptual summary of the extractors and reflectors. The specification environment is depicted on the left, showing the specification language UML and the tool, Poseidon for UML. The verification environment is depicted on the right, showing the verification languages, *LySa*, *PEPA* and *PEPA Nets*, and their associated tools, *PEPA Workbench*, *LySa Tool* and *PEPA Workbench for PEPA Nets*. Choreographer unifies the extractors, the reflectors, and the process algebra tools under a common hood, and then *choreographes* their interplay.

The LySa extractor requires class diagrams and a sequence diagram. The PEPA extractor looks for state diagrams, and a collaboration diagram, and the PEPA Net extractor looks for activity graphs.

The XMI menu has the following layout:

- Extract PEPA
- Extract LySa
- Extract PEPA Net
- Unzip Archive
- Refresh ZUML
- Generate LySa Narration
- Extract LySa from Diagram

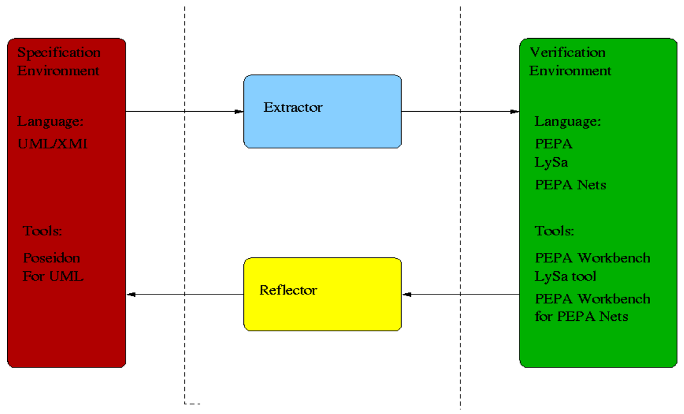

Specification Environment
Language: UML/XMI
Tools: Poseidon For UML
Extractor
Reflector
Verification Environment
Language: PEPA LySa PEPA Nets
Tools: PEPA Workbench LySa tool PEPA Workbench for PEPA Nets

Figure 10: Extractor/Reflector Concept

- Extract PEPA from Diagram
- Extract PEPA Net from Diagram

This menu is accessible from the XMI menu item in the menu-bar, and certain items are also accessible from the context menu for *zuml* and *xmi* files in the Explorer. *Zuml* files are recognised by the Explorer and displayed using the Poseidon Trident icon. *Xmi* files are recognised by the explorer and displayed using an *XMI* icon. Both can be expanded in the Explorer to show their contents. In particular, *xmi* files can be expanded to show a subset of the UML diagrams in the model. Choreographer will display UML activity graphs, sequence diagrams, and collaboration diagrams.

There are three other menu items, each called `Reflect`, in the PEPA, in the LySa and in the PEPA Net menu, that are related to Extraction/Reflection. These are used to invoke the reflectors for the respective process algebra, once the model in question has been solved.

The command `Extract PEPA` will launch the PEPA Extractor. This will take as input either a . *zuml* or *.xmi* file and produce a *.pepa* file of the same name.

The command `Extract LySa` will launch the For_LySa Extractor. This will take as input either a *.zuml* or *.xmi* file and produce a *.lysa* file of the same name.

The command `Extract PEPA Net` will launch the PEPA Net Extractor. This will take as input either a *.zuml* or *.xmi* file and produce a *.pepanet* file of the same name.

The command `Unzip Archive` will only work on *.zuml* files and has the effect of unzipping the archive to expose the *.xmi* file contained within.

The command `Refresh ZUML` can be used to refresh the *.zuml* file if it has been modified out-with Choreographer, for example by Poseidon.

The command `Generate LySa Narration` can be used to run the LySa extractor, and generates whats is known as a *Protocol Narration* for the given UML model.

The three commands `Extract LySa from diagram`, `Extract PEPA from diagram` and `Extract PEPA Net from Diagram` are used to run the extractors with a particular diagram as target. As mentioned before, *.xmi* files can be expanded in the Explorer to show the available diagrams in the model. The three actions to extract from a particular diagram are sensitive to the diagram type. `Extract LySa from Diagram` will only work for sequence diagrams, `Extract PEPA from Diagram` will only work for collaboration diagrams, and `Extract PEPA Net from Diagram` will only work for activity graphs. This was deemed a useful feature because real life UML models are likely to contain a variety of different diagram, for example multiple collaboration diagrams, where only one of these contains a suitable diagram for the extractor. Thus we have provided a way for you to select the correct diagram.

Xmi files may be opened in the Editor, syntax highlighting for them is available. The syntax highlighting engine cannot be customised due to certain limitations in NetBeans, and no background parsing is provided. *.zuml* files can be opened in the Explorer to show their contents, usually a project file containing Metadata and the *.xmi* file of the UML model itself. Both *.xmi* and *.zuml* files may be opened

For more information about the PEPA Extractor/Reflector, please see [12]. For more information about the LySa Extractor please see [13].

9 Acknowledgements

The author is supported by DEGAS (Design Environments for Global Applications) project, IST-2001-32072 funded by the FET Proactive Initiative on Global Computing.

References

[1] The DEGAS Website.
. http://www.omnys.it/degas/.

[2] PEPA Homepage, University of Ediburgh
. `http://www.lfcs.inf.ed.ac.uk/pepa/`.

[3] The LySa Homepage, Technical University of Denmark.
. `http://www.imm.dtu.dk/cs_LySa`.

[4] S. Gilmore, J. Hillston, M. Ribaudo, and L. Kloul. PEPA nets: A structured performance modelling formalism. *Performance Evaluation*, 54(2):79–104, October 2003.

[5] www.netbeans.org, homepage of the Netbeans IDE and Platform
. `http://www.netbeans.org/`.

[6] Poseidon for UML Website, Gentleware AG, Hamburg, Germany
. `http://www.gentleware.com/`.

[7] Sun Microsystems, The Java Programming Language
. `http://java.sun.com/`.

[8] Standard ML of New Jersey
. `http://www.smlnj.org/`.

[9] DEGAS Choreographer Website, University of Edinburgh
. `http://groups.inf.ed.ac.uk/choreographer/`.

[10] PEPA Workbench, Java Edition, University of Edinburgh
. `http://homepages.inf.ed.ac.uk/s9905941/jPEPA/`.

[11] The LySa analyser developed by Mikael Buchholtz, Techincal University of Denmark.
. `http://www.imm.dtu.dk/cs_LySa/lysatool`.

[12] C. Canevet, S. Gilmore, J. Hillston, M. Prowse, and P. Stevens. Performance modelling with UML and stochastic process algebras. *IEE Proceedings: Computers and Digital Techniques*, 150(2):107–120, March 2003.

[13] M. Buchholtz, C. Montangero, L. Perrone, and S. Semprini. For-lysa: UML for authentication analysis. In *Proceedings of the second workshop on Global Computing*, Lecture Notes in Computer Science. Springer Verlag, 2004.