SpinRCP – The Eclipse Rich Client Platform Integrated Development Environment for the Spin Model Checker

Zmago Brezočnik, Boštjan Vlaovič, and Aleksander Vreže

Faculty of Electrical Engineering and Computer Science,
University of Maribor, Slovenia
{zmago.brezocnik, bostjan.vlaovic, aleksander.vreze}@um.si

Abstract. SpinRCP is an integrated development environment (IDE) for the Spin model checker that is used for verifying the correctness of concurrent and distributed systems. Using SpinRCP, it is easy to enter, edit, examine, and check syntax of the models that represent the systems to be analyzed, to check redundancies in models, to specify the required properties of models, to graphically represent processes and never claims derived from the specified properties in the form of nondeterministic final state machines, to enter or select many various simulation and verification parameters, to perform verification and random, guided, or interactive simulation and to transform a Spin simulation trail to a standard Message Sequence Chart (MSC). SpinRCP is implemented in Java as an Eclipse Rich Client Platform (RCP) product.

Keywords: Model Checking, Integrated Development Environment, Spin, SpinRCP, Eclipse Rich Client Platform

1 Introduction

The sizes and complexities of modern ICT systems and applications are increasing rapidly. On the other hand, designers are faced with demands for reduction in development costs and shorter time-to-market for a new product. Additional difficulties for them stem from the fact that distributed, parallel, or multithreaded programs run concurrently. Fortunately, powerful formal methods and tools are yet available that can verify the logical correctness of concurrent software. Of one the most successful method that is being used for the verification of real-life systems not only in academia but also in an industrial context is that of model checking. Perhaps the most widespread and useful model checking tool is Spin [5]. It is a command line tool. A user enters the commands at the command line and Spin outputs the results there. Such an approach to model checking could be difficult especially for newcomers who are not yet well acquainted with Spin commands. Of course, the availability of a user-friendly graphical interface to Spin is of a considerable benefit for skilled users as well. In order to accomplish these needs, several graphical interfaces or environments for Spin have been developed.

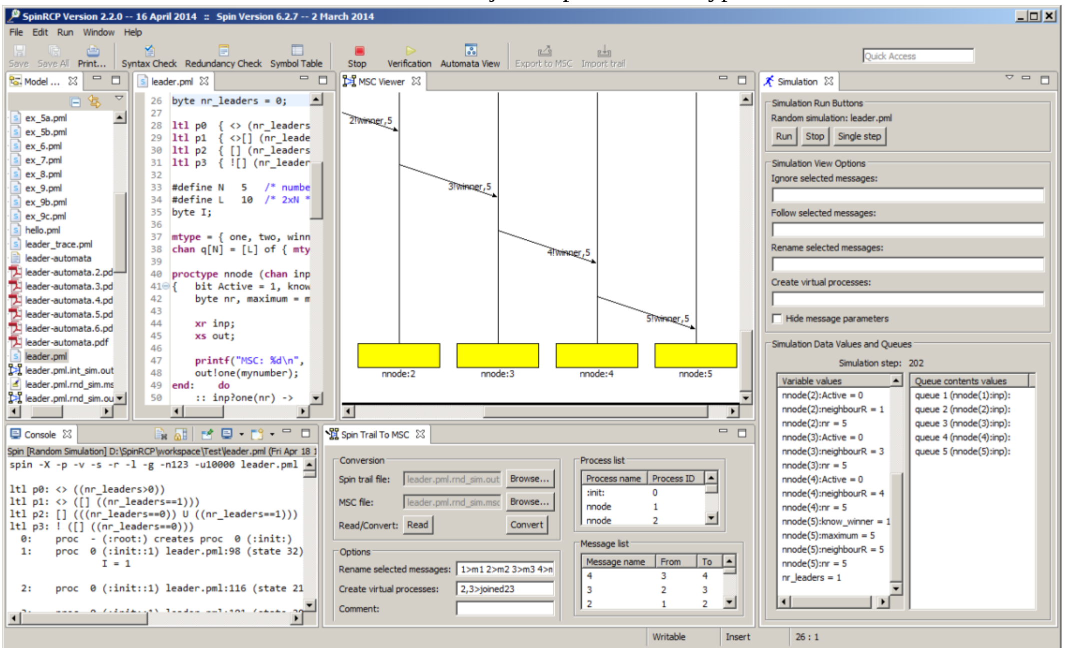
Xspin [5] was the first graphical interface to Spin. It was written in Tcl/Tk script language. The last version of Xspin was Version 5.2.5 from 17th April 2010. Since then it has no longer been supported. jSpin [1] is an alternative graphical user interface for Spin. It was developed by M. Ben-Ari together with its SpinSpider component primarily for demonstrating the properties of concurrent programming. It is written in Java. Motivated by a need for a capable Promela editor when we introduced our sdl2msc tool [9, 10] that generated a huge Promela model of an important part of the Iskratel SI3000 Softswitch specified in SDL, we developed an Eclipse Plug-in for Spin in Java [7]. iSpin [6] is the graphical user interface that has replaced Xspin since Spin Version 6.0.0. It was provided by the Spin author, G. J. Holzman, and is included in each new Spin release. Just like Xspin, iSpin is implemented using the Tcl programming language and the Tk graphical user interface toolkit. EpiSpin [3], introduced by de Vos et. al., is another Eclipse plug-in for editing Promela models and starting Spin verification and simulation runs. Still another tool called COMPL$_e$T$_e$ [4] is used within the Eclipse. It is actually a toolchain for validation of communication protocols that combines the possibility of an abstract behavior description represented as UML-Statechart models with a formal representation in Promela, which is used by Spin. In this paper we introduce our new integrated development environment for Spin called SpinRCP [8, 2].

The structure and main functionalities of the SpinRCP IDE are presented in Section 2. Section 3 draws together its most important features and gives some ideas and plans for further work.

2 Structure and main functionalities of SpinRCP

Based on the experiences gained in the development of Eclipse Plug-in for Spin [7], we decided to develop an integrated development environment for Spin users, which will facilitate editing and reviewing large Promela models including those extracted from an existing software code, simple parameters choosing for individual operations on the model, running Spin verification and simulation, graphical display of MSCs, and keeping records of file versions. For the implementation of this environment, we selected the Eclipse Rich Client Platform (RCP) technology. RCP is the minimum set of plug-ins needed to build a rich client application. It allows us to quickly build a professional-looking application, with native look-and-feel, on multiple platforms. The application opens in a window called a Workbench that is entitled with SpinRCP and Spin versions and release dates, respectively. SpinRCP Workbench contains more perspectives. Each perspective contains parts (views and editors) and controls what appears in certain menus and tool bars. In SpinRCP, there are 19 different views (e.g. Model Navigator, Console, Simulation, Spin Trail To MSC, Help, CVS Repositories...) and only two editors (Promela Editor and MSC Viewer). Using a simple drag-and-drop operation you can relocate and/or resize any part and thus reform the perspective at your will. Particular perspectives can be saved and later opened when needed. The outlook of Workbench with Spin RCP perspective with the Menu bar, the Tool bar, the Model Navigator View, the Console View, the Promela Editor, the MSC Viewer Editor, and the Spin Trail To MSC View is shown in Fig. 1. SpinRCP needs four external tools: Spin, C compiler, Java, and Graphviz dot. Paths to those tools must be set in the Spin preference page. For ease of use, it is advised to set also some useful options in the General preference page. Promela models can be either created or imported into the workspace in the Model Navigator view. Models in the workspace may be organized in tree-like folder/subfolder hierarchy. If you have an existing Promela model anywhere in your file system, you can simply copy-paste or drag-and-drop it to the appropriate place in the workspace. For ease of viewing and editing models the following features are available: syntax highlighting, code folding, content assist, and marking a place of a syntax error.

If a model in the Promela Editor is selected, several Tool bar icons for launching a specific SpinRCP action become enabled. Syntax Check uses Spin –a option for performing a thorough model syntax check and generates the source C program for a model-specific verifier. If the Spin syntax checker detects an error within Promela source code, SpinRCP marks the line where the error occurred with an error icon. Redundancy Check uses Spin –A option to apply a property-based slicing algorithm for the model, which can detect eventual redundancies in the model and generate suggestions on how the model could be revised in order to use less memory. Symbol Table uses the Spin –d option to produce symbol table information for the Promela model. The information for each Promela object depends on its type.



```
byte nr_leaders = 0;

ltl p0  { <> (nr_leaders
ltl p1  { <>[] (nr_leade
ltl p2  { [] (nr_leaders
ltl p3  { ![] (nr_leader

#define N   5   /* numbe
#define L   10  /* 2xN *

byte I;

mtype = { one, two, winn
chan q[N] = [L] of { mty

proctype nnode (chan inp
{   bit Active = 1, know
    byte nr, maximum = m

    xr inp;
    xs out;

    printf("MSC: %d\n",
    out!one(mynumber);
end:    do
    :: inp?one(nr) ->
```

Spin [Random Simulation] D:\SpinRCP\workspace\Test\leader.pml (Fri Apr 18 1

```
spin -X -p -v -s -r -l -g -n123 -u10000 leader.pml

ltl p0: <> ((nr_leaders>0))
ltl p1: <> ([] ((nr_leaders==1)))
ltl p2: [] (((nr_leaders==0)) U ((nr_leaders==1)))
ltl p3: ! ([] ((nr_leaders==0)))
  0:    proc  - (:root:) creates proc  0 (:init:)
  1:    proc  0 (:init::1) leader.pml:98 (state 32)
                I = 1


  2:    proc  0 (:init::1) leader.pml:116 (state 21
```

Conversion
Spin trail file: leader.pml.rnd_sim.out [Browse...]
MSC file: leader.pml.rnd_sim.msc [Browse...]
Read/Convert: [Read] [Convert]

Options
Rename selected messages: 1>m1 2>m2 3>m3 4>n
Create virtual processes: 2,3>joined23
Comment:

Process list

Process name	Process ID
:init:	0
nnode	1
nnode	2

Message list

Message name	From	To
4	3	4
3	2	3
2	1	2

Simulation Run Buttons
Random simulation: leader.pml
[Run] [Stop] [Single step]

Simulation View Options
Ignore selected messages:
Follow selected messages:
Rename selected messages:
Create virtual processes:
☐ Hide message parameters

Simulation Data Values and Queues
Simulation step: 202

Variable values	Queue contents values
nnode(2):Active = 0	queue 1 (nnode(1):inp):
nnode(2):neighbourR = 1	queue 2 (nnode(2):inp):
nnode(2):nr = 5	queue 3 (nnode(3):inp):
nnode(3):Active = 0	queue 4 (nnode(4):inp):
nnode(3):neighbourR = 3	queue 5 (nnode(5):inp):
nnode(3):nr = 5	
nnode(4):Active = 0	
nnode(4):neighbourR = 4	
nnode(4):nr = 5	
nnode(5):know_winner = 1	
nnode(5):maximum = 5	
nnode(5):neighbourR = 5	
nnode(5):nr = 5	
nr_leaders = 1	

Fig. 1. SpinRCP Workbench perspective with four different views and two editors

Automata View opens the Automata View preference page, where the user can select in which graphical format the automata should be displayed. Ten different file formats are currently available. SpinRCP uses options –o3 and –a to generate the verifier source C code, then compiles it to pan and runs pan using run-time option –D. This option generates state tables for each proctype and each never claim in the format accepted by the dot tool from Graphviz. These state tables are redirected to a text file. Next, the dot tool transforms this text file to a set of files (one file for automaton) with the previously selected graphical format. Then, a dialogue appears, where the automaton to be displayed can be selected. Finally, a system program, assigned to a given file type is opened and the selected automaton is displayed.

Verification opens the Verification preference page. In the upper part, the user can export current verification parameters (verification profile) to an xml file and import or reload a previously saved verification profile. Verification options are accessible below in two tabs: Basic Options and Advanced Options. In the Basic Options tab, a user can select a correctness property to be proved with several additional options, the search mode, a full queue behavior during verification, the explicit use of user-entered compile-time and run-time parameters that supersede the clicked options and the elsewhere entered parameters, and how a never claim (if any) is specified: with the in-model LTL formula/claim name, an LTL formula in the text field, an LTL formula in a 1-line file, and never claim in a file. In the Advanced Options tab it is possible to enter several advanced verification parameters and select some error-trapping and verification run type options. If the verification finds an error, a counterexample is generated and saved in a so-called error-trail file with the file extension .trail appended to the original Promela model file name. The trail file can be used in a guided simulation that replays the execution that violated the property.

Simulation opens the Simulation preference page, where the user can select the type of simulation (random, guided, interactive), the number of initial steps skipped, the maximum number of simulation steps, how a full queue is simulated (either blocks or loses new messages), a seed value for the random simulation, and the trail file for the guided simulation. Then the Simulation view opens and the simulation can start. Interactive simulation allows a user to select one out of all executable Promela statements at a given moment in the interactive simulation dialogue in order to resolve the non-deterministic continuation of the execution run. Guided simulation uses an error-trail file produced in a verification run. Therefore, if the verification discloses an error, the guided simulation will be used for displaying the execution run that violates the checked property of a model. During the simulation, two Java threads run in parallel: a Spin simulation thread and an MSC refreshing thread. The simulation thread executes the Spin simulation and displays Spin textual simulation outputs on the Console. In parallel, the MSC refreshing thread is displaying the MSC in MSC Viewer. Currently, continuous and message-by-message simulation is available. The variable values and queue contents values are updated in two separate tables at the bottom of the Simulation View. In order to adapt the display of MSCs, the user can either select or deselect particular messages in an MSC, rename selected messages and join processes into a new virtual process and select whether to show or hide message parameters. SpinRCP includes another view called Spin Trail To MSC that is intended for converting a Spin simulation output file (out) to the standard MSC text file (msc) formatted according to ITU-T Z.120. As for the viewing a MSC in the MSC View, the same options are available to be set before the conversion. The abstraction using the introduction of virtual processes is very useful for analyzing huge MSCs. It hides details that are not interesting at a given abstraction level and would otherwise represent an unnecessary complexity.

3 Implementation Details

SpinRCP is written completely in Java within the Eclipse IDE and then exported as an Eclipse Rich Client Platform (RCP) product. Therefore, it runs as a stand-alone RCP application without the need to have Eclipse IDE installed, because all required plug-ins from Eclipse are already bundled in SpinRCP. The whole SpinRCP product consists of 122 plug-ins. One of them is our plug-in called org.um.feri.spin.rcp, which contains the total amount of more than 16,000 lines of Java source code in 19 Java packages with a total of 84 files defining Java classes. The help contents for SpinRCP is implemented in a separate plug-in that contains more than 60 html files with descriptions of individual help topics and many xml configuration files. In addition, SpinRCP uses a slightly modified version of the st2msc Java application [7] integrated as an internal jar file. SpinRCP is freely available at the website http://lms.uni-mb.si/spinrcp.

There you can find links to the required external software, download and installation instructions, SpinRCP release notes, a brief user guide through all functionalities, some references and answers to frequently asked questions. Currently, SpinRCP is provided for 32- and 64-bit Windows operating systems. Since special attention was given to write platform independed code, it will be released also for Unix/Linux and Mac OS X platforms after a thorough testing. It is compatible with all Spin versions, including those before Spin Version 6.0.0 that have different linenumber/filename references. Of course, some options might not work with older versions (e.g., the Automata View is feasible only from Spin Version 6.0.0).

4 Conclusion

The most important features of SpinRCP are the following ones: a user-friendly Promela editor with syntax coloring, code folding, keyword autocompletion, and syntax error marking, running Spin verification, random, guided, and interactive simulation, graphical MSC viewing, abstracting MSCs by joining some processes into an abstract process, conversion of Spin simulation output to a standard text file, which is readable by external MSC viewers, displaying graphical automata representation of proctype definitions and never claims in a model in different graphical formats. We have many ideas and plans for further improvements and new features of SpinRCP: options for filtering the Spin simulation output to console, a rewind option together with stepping forward and backward in step simulation, indication of the statement that is currently executed during a simulation run in the Promela source file, cleanup of temporary files, display of a process creation in the MSC Viewer, a command to display the state tables for proctype definitions and never claims, an extension of the Automata view that displays the FSMs of proctype definitions and never claims of a corresponding Promela model by highlighting the visited states in the guided simulation run that illustrates the violation of the property being checked, Spin swarm support for distributing a model checking task to more CPU cores or to a cloud of workstations, generation of verification reports in textual, tabular, and/or graphical form, etc.

References

1. Ben-Ari, M.: jSpin – Java GUI for SPIN: User's Guide, Version 5.0, (2010)
2. Brezočnik, Z., Vlaovič, B., Vreže, A.: Model Checking using Spin and SpinRCP. Informacije MIDEM, Journal of Microelectronics, Electronic Components and Materials, vol. 43, no. 4, pp. 235-248, (2013). Electronic version available at http://www.midem-drustvo.si/Journal/Search.aspx?Vol=43&Iss=4
3. de Vos, B., Kats, L.C.L., Pronk, C.: EpiSpin: An Eclipse Plug-In for Promela/Spin Using Spoofax, In: Groce, A., Musuvathi, M. (eds.) SPIN 2011. LNCS, vol. 6823, pp. 177- 182. Springer, Heidelberg (2011)
4. Gröning, S., Rosas, C., Wietfeld, C.: COMPL$_e$T$_e$ – A COMmunication protocol validation Toolchain using Formal and Model-Based Specifications and Descriptions. In: Bartocci, E., Ramakrishnan C.R. (eds.): SPIN 2013, LNCS 7976, pp. 18–23. Springer, Heidelberg (2013)
5. Holzmann, G.J.: The Spin Model Checker: Primer and Reference Manual. Addison-Wesley, Boston (2004)
6. Hornos, M.J., Augusto, J.C.: Installation Process and Main Functionalities of the Spin Model Checker. Electronic version available at http://digibug.ugr.es/handle/10481/19601 (2012)
7. Kovše, T., Vlaovič, B., Vreže, A., Brezočnik, Z.: Eclipse plug-in for spin and st2msc tools - tool presentation. In: Păsăreanu, C.S. (ed.) SPIN 2009. LNCS, vol. 5578, pp. 143- 147. Springer, Heidelberg (2009)
8. Kovše, T.: Environment for formal verification of safety-critical systems, Master Thesis (in Slovene), Faculty of EE&CS, University of Maribor, Slovenia (2011)
9. Vlaovič, B., Vreže, A., Brezočnik, Z., Kapus, T.: Automated generation of Promela model from SDL specification. Computer Standards and Interfaces, vol. 29, no. 4, pp. 449- 461 (2007)
10. Vreže, A., Vlaovič, B., Brezočnik, Z.: Sdl2pml - tool for automated generation of Promela model from SDL specification. Computer Standards and Interfaces, vol. 31, no. 4, pp. 779- 786 (2009)

A. Oral Tool Presentation

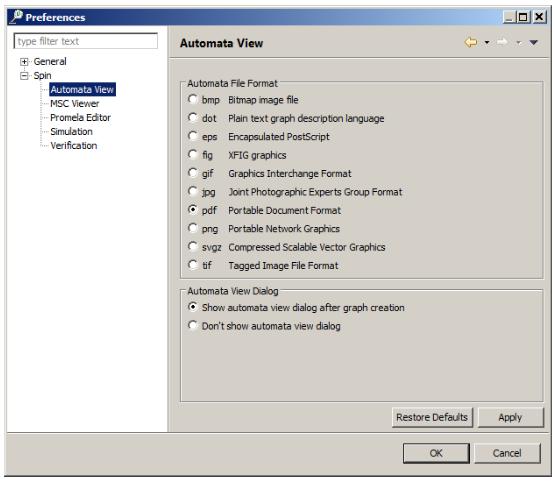
This section gives a plan for oral presentation of the SpinRCP tool.

A.1 The concept and structure of SpinRCP

First, the motivation for designing an integrated development environment for the Spin users as a stand-alone Eclipse Rich Client Platform product is explained and followed with a brief presentation of its structure.

A.2 Live demonstration of SpinRCP functionalities

SpinRCP functionalities are demonstrated live on the leader.pml model from the Test folder in standard Spin distributions. Fig. A.1 shows some capabilities in the Automata View option and interactive simulation.

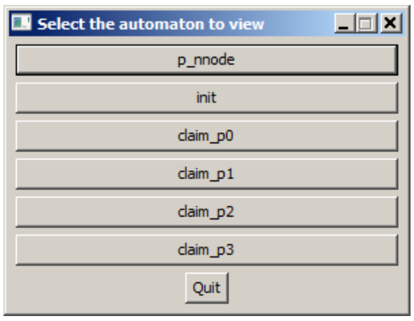


Automata View

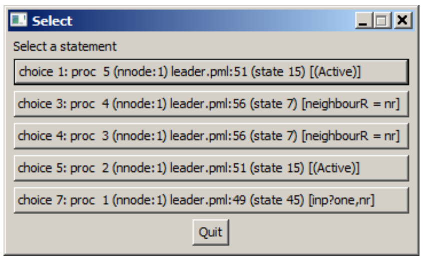
Automata File Format
- bmp Bitmap image file
- dot Plain text graph description language
- eps Encapsulated PostScript
- fig XFIG graphics
- gif Graphics Interchange Format
- jpg Joint Photographic Experts Group Format
- [✓ pdf Portable Document Format]
- png Portable Network Graphics
- svgz Compressed Scalable Vector Graphics
- tif Tagged Image File Format

Automata View Dialog
- [✓ Show automata view dialog after graph creation]
- Don't show automata view dialog

a) Automata View preference page



b) Select the automaton to view dialogue



Select a statement

choice 1: proc 5 (nnode:1) leader.pml:51 (state 15) [(Active)]

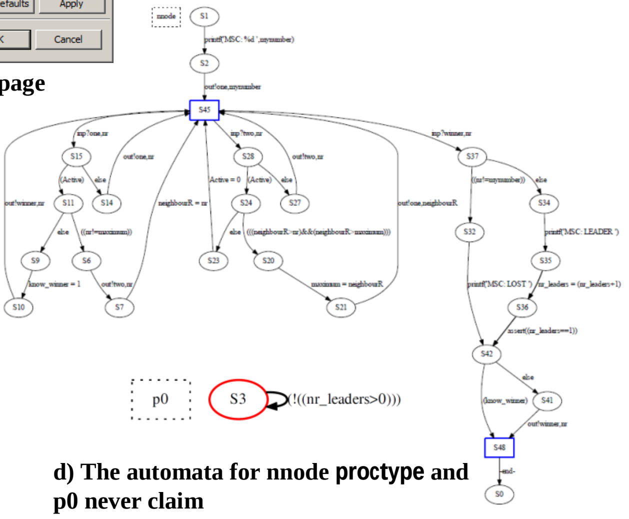
choice 3: proc 4 (nnode:1) leader.pml:56 (state 7) [neighbourR = nr]

choice 4: proc 3 (nnode:1) leader.pml:56 (state 7) [neighbourR = nr]

choice 5: proc 2 (nnode:1) leader.pml:51 (state 15) [(Active)]

choice 7: proc 1 (nnode:1) leader.pml:49 (state 45) [inp?one,nr]

c) Interactive simulation dialogue



d) The automata for nnode proctype and p0 never claim

Fig. A.1. Some capabilities of SpinRCP in Automata View option and interactive simulation

Finally, we present our experience with the product when dealing with two Promela models automatically extracted from the SDL code. The first one is the ITU-T V.76 protocol, whose extracted Promela model has more than 4,600 lines. The second one is the IUA protocol (ISDN User Adaptation Layer Protocol) implemented in a real product with a huge extracted model of more than 102,000 lines of generated Promela code.

SpinRCP website: http://lms.uni-mb.si/spinrcp

There you can find links to the required external software, download and installation instructions, SpinRCP release notes, a brief user guide through all functionalities, SpinRCP and related references, and answers to some frequently asked questions.