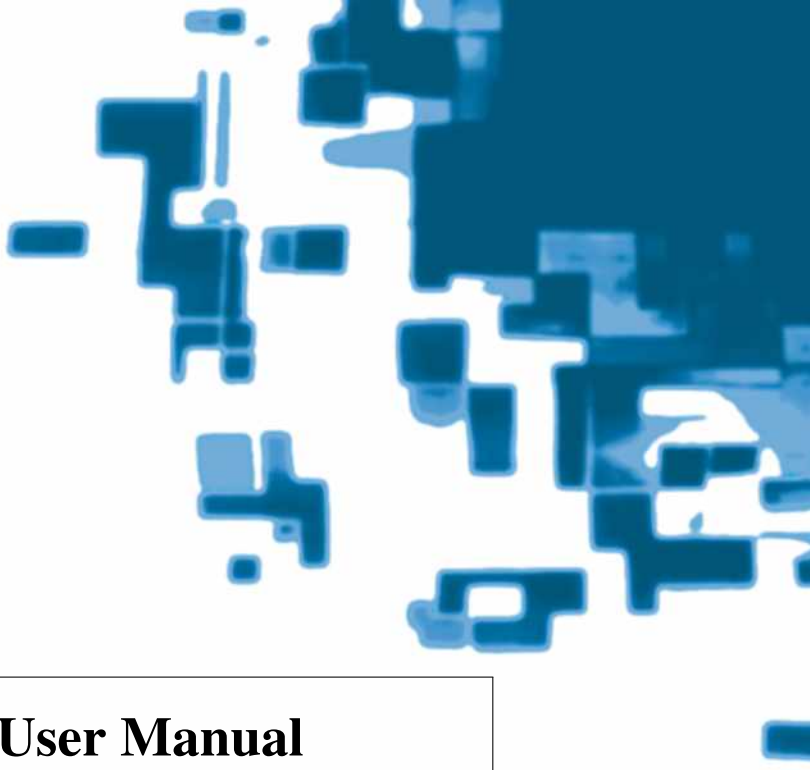

*S*BIP 2.0: User Manual

Braham Lotfi Mediouni, Ayoub Nouri, et al.

Verimag Research Report n^{o} xxx

April 27, 2018

Reports are downloadable at the following address
http://www-verimag.imag.fr

Unité Mixte de Recherche 5104 CNRS - Grenoble INP - UGA
Bâtiment IMAG
Université Grenoble Alpes
700, avenue centrale
38401 Saint Martin d'Hères
France
tel : +33 4 57 42 22 42
fax : +33 4 57 42 22 22
http://www-verimag.imag.fr/

$\mathcal{S}$BIP 2.0: User Manual

Braham Lotfi Mediouni, Ayoub Nouri, et al.

April 27, 2018

Abstract

This document is a user manual for $\mathcal{S}$BIP 2.0, a Statistical Model Checker

Keywords: Stochastic Models, Statistical Analysis, Automated Verification

Reviewers: Marius Bozga, Axel Legay and Saddek Bensalem

How to cite this report:

```
@techreport {xxx,
  title = {SBIP 2.0: User Manual},
  author = {Braham Lotfi Mediouni, Ayoub Nouri, et al.},
  institution = {{Verimag} Research Report},
  number = {xxx},
  year = {}
}
```

Contents

1 Introduction

$\mathcal{S}$BIP 2.0 [1] is an IDE including project management, model edition, compilation, simulation, and statistical analysis for BIP models. It is based on the BIP framework for modeling in a component-based fashion. The current release allows using different BIP engines, namely, untimed BIP, and the new Stochastic Real-Time BIP.

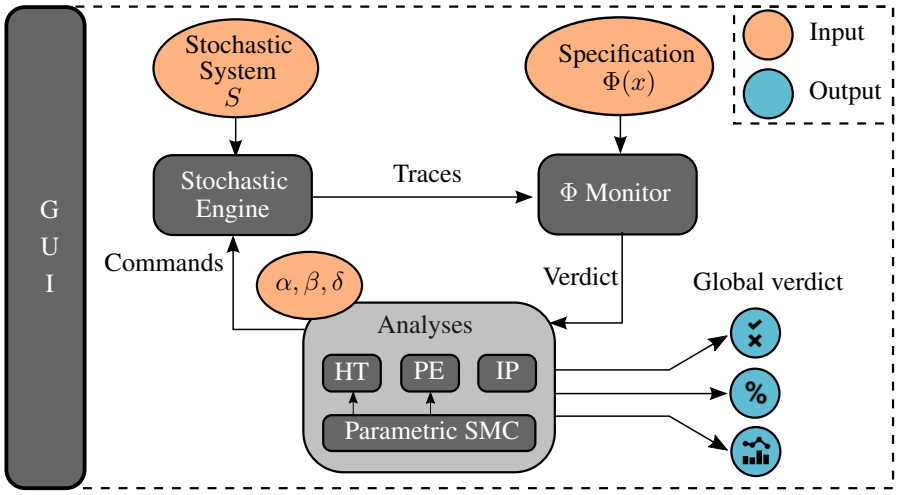

Figure 1: $\mathcal{S}$BIP architecture

The tool relies on five modules, namely, a Stochastic Simulation Engine, a Monitoring module, an SMC Engine, a Rare-Event engine, a Parametric Exploration module plus additional data structures. The stochastic engine encapsulates the behavior of an executable model simulator and is used to produce (random) execution traces on demand. The monitor is used to evaluate properties on a trace. The SMC engine implements the main statistical-model-checking loop depending on the statistical method used, namely hypothesis testing or probability estimation. The rare-event engine implements the Importance Splitting technique to estimate the probability of rare properties. Finally, the parametric exploration module coordinates the evaluation of a parametric property.

This tool is available online [2] in different artifacts: the source code, the installable artifact and a pre-configured virtual machine. To avoid setup issues we advise using the virtual machine that is pre-configured to make the tool run properly.

[1] Details about the tool can be found at: `address.of.the.technical.report`
[2] http://www-verimag.imag.fr/BIP-SMC-A-Statistical-Model-Checking.html

2 Application setup

The tool and the bip engines were developed using Java 7 and work only on GNU/Linux OS. The Java Runtime Environment 7 is available for download online [3]. After installing java 7, make sure that it is your default java.

In this section, we explain how to install the $\mathcal{S}$BIP 2.0, the GNU scientific library and finally the BIP engines.

2.1 $\mathcal{S}$BIP 2.0 setup

The tool is released with an installation file named 'install.sh'. To install the tool, open a terminal and run the following command :

```
$ ./install.sh
```

The installation generates a desktop launcher that allows to start the tool by double-clicking on the generated launcher.

2.2 GNU Scientific Library setup

GSL is a numerical library that we use to sample probability density functions. It is released under a GNU General Public License. This library can be downloaded online [4] and installed by running the command below in the downloaded folder with the required priviliges:

```
$./configure; make; make install
```

NB: In some cases, an error may happen while loading the library, checking for 'libgsl.so.0' instead of 'libgsl.so'. This can be solved by creating the following link:

```
$ ln /usr/local/lib/libgsl.so /usr/local/lib/libgsl.so.0
```

2.3 Simulation engines setup

The BIP engines are provided by default in all the distributed versions of the tool. This installation is useful in the case the user wants to donwload a more recent (or older) version of the BIP engines and use them in $\mathcal{S}$BIP . The engines are available for download online:

- Untimed BIP :
  http://www-verimag.imag.fr/New-BIP-tools.html
- Stochastic Real-Time BIP :
  http://www-verimag.imag.fr/BIP-SMC-A-Statistical-Model-Checking.html

Note that BIP model compilation requires to install the packets g++ and cmake. In a terminal, run the command :

```
$ sudo apt-get install g++ cmake
```

After downloading the engines, one has to configure the BIP compilation in $\mathcal{S}$BIP . This consists of updating the compilation scripts by locating the installed BIP engines. To configure the Stochastic RT-BIP engine for example, follow these steps:

1. Open $\mathcal{S}$BIP by clicking on the launcher
2. In the toolbar, click on the button 'Configure BIP compilation scripts'.

[3] Java 7 download link: http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html

[4] GSL download link: ftp://ftp.gnu.org/gnu/gsl/

3. In the dialog window that opens, select 'Stochastic RT-BIP' in the combo box, then click 'ok'. This will open an edition tab that contains the corresponding script.
4. At the line preceding the line *source setup.sh reference-engine*, update the current cd line so that it points to the folder 'distribution' of the BIP engine you installed. The result should be :

```
cd /path/to/bip/engine/folder/distribution
source  setup.sh reference-engine
```

5. Save the changes
6. Close the edition tab

3 Graphical User Interface

The GUI is organized in three main regions : (1) a project explorer, (2) a toolbar, and (3) a central panel. The project explorer organizes different files in a tree hierarchy. The Models folder contains (.bip) models, auto-generated and external (.cpp/hpp) source code, custom probability distributions, and executables. The Properties folder stores (.mtl) and (.ltl) properties in addition to (.sf) scoring functions. Finally, the Outputs folder contains execution traces.

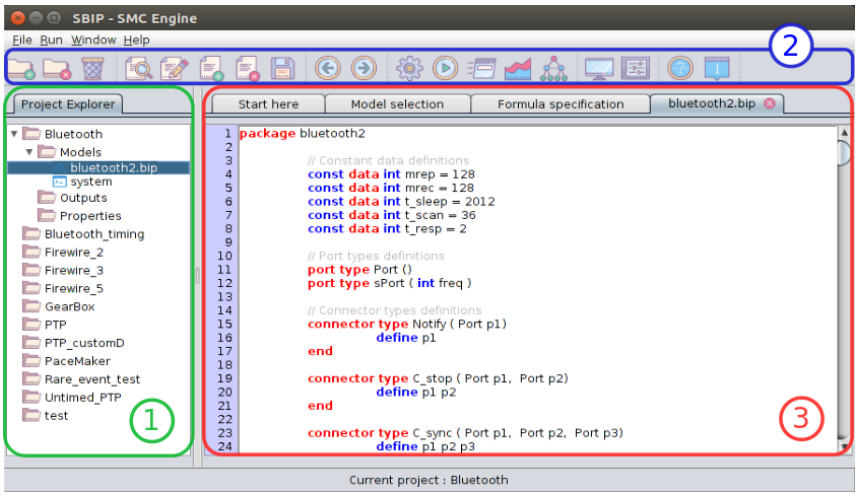

Figure 2: Screen-shot of $\mathcal{S}$BIP graphical user interface

The toolbar is organized in six functional groups: (i) project management, allowing to create/remove projects, (ii) file management for model files creation, deletion, edition and visualization, (iii) tab navigation, (iv) workflow management for model compilation, simulation and analysis, (v) configuration setup, and (vi) help buttons.

The central panel provides several views:

- edition view: used to edit models and properties.
- configuration view: used to select the simulation engine, the SMC algorithm and parameters, the instantiation domain for parametric properties,
- analysis view: used to initiate and track the progress of analysis,
- results view: used to provide a summary of the performed analysis.

4 Model Edition

4.1 BIP framework

BIP stands for Behavior-Interaction-Priority and provides concepts for modeling and design of heterogeneous component-based systems using a layered approach. In BIP, systems are obtained by composition of atomic components (i.e., the behavior) with multiparty hierarchical interactions, and coordinated using dynamic priorities. BIP was mainly targeted for rigorous design of component-based systems, that is, not only formal modeling and analysis but also correct-by-construction implementation and deployment. As such, BIP offers facilities to incorporate and execute external code within atomic components and/or multiparty interactions. The documentation of BIP 2 is available online [5].

4.2 Stochastic extension

The Stochastic Real-Time BIP extension supports a new type of interaction, namely, stochastic. These interactions are defined using specific annotations @stochastic(dist="...", clk=..., param="...") to tag components ports. Such annotations specify a probability density function and its parameters through, respectively, the $dist$ and $param$ attributes, and to associate a clock through the clk attribute. Currently, the language supports a number of built-in density functions : 'normal', 'gamma' and 'chi_squared'. Note that 'uniform' and 'exponential' distributions can also be specified in the annotations.

Additional functions can be used through the same mechanism by defining a 'custom' distribution in the $dist$ attribute, and in $param$ a file characterizing the underlying cumulative distribution. An example of such a file can be found in the case study 'PTP'.

Details about the stochastic semantics of RT-BIP can be found online [6].

4.3 Model importation

Separately designed models can be easily imported to a design project. This is applicable to all kinds of modeling files (.bip/cpp/hpp/txt) but also compiled models (usually named 'system'). To do so:

1. Select the project in the project exploration panel,
2. In the central panel, select the 'model selection' tab,
3. In the file chooser, select the file to import,
4. Click on the button 'open',
5. The file is added to the 'Models' folder of the corresponding project.

4.4 Model edition

A model file can be visualized by double-clicking on it in the project exploration panel, or by selecting it and clicking the 'Edit file' button. This file is displayed in a dedicated edition tab, added to the central panel. The edition view provides facilities for BIP models design, that are :

- Keyword highlighting for the BIP 2 syntax
- Code auto-completion - This functionality allows to insert code patterns for a : package, atom type, compound type, etc. This is activated by using the CTRL+SPACE keys. For example, type pac in the edition tab then type CTRL+SPACE, and finally select 'create package' in the popup menu.
- Find/Replace option using CTRL+F. Note that recursive replacements lead to infinite loops. eg. replacing 'do' by 'do this'.

[5] BIP documentation: http://www-verimag.imag.fr/TOOLS/DCS/bip/doc/latest/pdf/BIP2.pdf
[6] BIP documentation: http://www-verimag.imag.fr/TR/TR-2017-6.pdf

- Multi-line commenting/uncommenting. Select several lines of code then press CTRL+SHIFT+C. This comments the uncommented selected lines, and vice versa.
- Double-click highlighting. In the code, double-click on a word. This highlights the word in blue and all its occurrences in yellow. This can be useful to track a variable during the development phase.
- Undo/Redo using respectively CTRL+Z and CTRL+Y.
- Multi-line tab/untab. Select several lines of code then press CTRL+T to tab all the lines and CTRL+SHIFT+T to untab the selected lines that are preceded by a tab.

After editing a file one has to make sure to save it before closing the tab. We also recommand to frequently save your changes.

4.5 Compilation

The compilation of a BIP model produces a 'system' file in the 'Models' folder. The system is a runnable file that can be used in simulation and analyses. To compile the BIP model, first open the edited model then click on the 'Compile BIP model' button. After, select the type of BIP compiler and enter the name of the package and compound. A dialog window opens and shows the progression of the compilation, including warnings and potential errors.

The compilation operation generates a 'system' file. Note that if a 'system' file already exists, the compilation overwrites the existing file. Also note that the BIP model must be saved before compiling in order to take into account the last changes.

4.6 Simulation

To simulate a system, first select it in the project explorator. The URL of the selected system is loaded in the 'Model selection' tab. Then, click on the 'Simulate system' button in the toolbox. A simulation tab opens in the central panel. Simulation parameters can be specified. Press 'Start simulation' to start the simulation. Finally, you can stop then reset simulation by clicking the corresponding button.

For more information about the available parameters, you can run a sumilation with the parameter

`--help`.

5 Properties edition

In this section we detail the MTL properties features. Note that the LTL properties are manipulated in a similar manner since LTL and MTL both fulfill the same requirements described by well-defined Java interfaces.

5.1 the MTL monitor

5.1.1 The architecture

The MTL monitor, implements an online monitoring algorithm based on predefined rewriting rules. Given an MTL formula ϕ and a timed trace ω, the monitor alternates rewriting and simplification phases. Rewriting consumes a symbol σ of ω and partially evaluates the current formula ϕ into ϕ'. Partial evaluation includes the unfolding of temporal operators and evaluation of atomic state formula to their truth value. Simplification aims at applying reduction rules on the formula ϕ' based on Boolean logic.

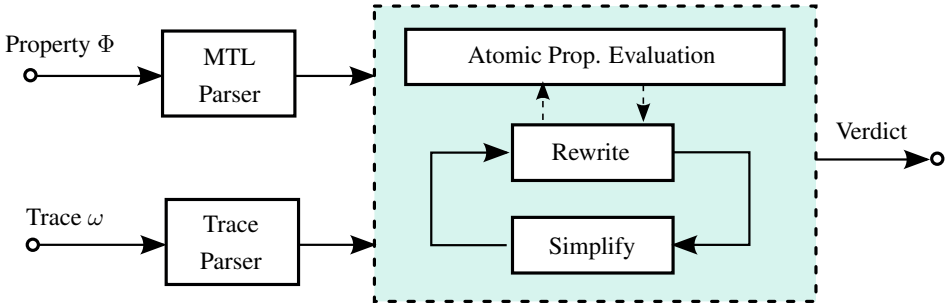

Figure 3: Functional view of the MTL Monitor

5.1.2 The syntax

Here we detail the syntax of an MTL formula:

- Temporal operators : Until (U), Release (R), Globally (G), Eventually (F) and Next (N)
- Boolean operators : And (&&), Or (||) and Not (!)
- All temporal operators (except the Next operator) are followed by an interval
- Intervals are defined as follows : "[lower_bound, upper_bound]"
- State formulae are boolean expressions over data variables
- Arithmetic operations can be used in the boolean expressions
- Available mathematical functions : exponential (exp) and absolute value (abs)
- The formula must be written in Negative Normal Form

For more clarity, we advise to use parentheses when writing properties. Note that the right operand must be between parentheses in the case of And operator.

5.1.3 The edition

To edit a property, select the current project, then go to the 'formula specification' tab and type your formula. Once you are done, you can check the syntactic validity the property and/or save the property in the 'Properties' folder. Saving properties allows to keep track on the verified properties and to load them easily for further analyses.

NB : if you save a property with an existing name, it overwrites the existing one.

5.2 Parametric properties

A parametric property $\phi(x)$ is a property that contains a parameter variable. This variable is used for a parametric exploration. This parameter is not only restricted to the variable name 'x', as long as the same name is specified during the parametric exploration.

6 System analysis

6.1 The SMC Engine

The SMC engine implements several statistical testing algorithms for stochastic systems verification, namely, Single Sampling Plan (SSP), Simple Probability Ratio Test (SPRT) for hypothesis testing, and Probability Estimation (PESTIM). To run an SMC, follow these steps:

1. Select a 'system' file
2. Select a non-parametric property or edit a new one.
3. Note that the two files must be in the same project.
4. Launch an SMC by clicking on the button in the toolbar
5. Set all the analysis parameters
6. Start the analysis by pressing the button 'Start simulation'
7. A dialog window appears and shows the progression of the simulation
8. At the end a result tab is added to the central panel containing a summary on the analysis

Note that the simulation can be aborted by closing the dialog window. Also, the other tabs are disabled during the result display.

The execution traces can be generated in two modes: symbol-by-symbol or at-once. At-once generation takes as extra parameter the maximum trace length which can make the monitor deal with traces that are too short. In this case, the monitor returns a 'not able to conclude' verdict. In order to make the SMC core deal with non-informative traces, these verdicts are pessimistically considered as prefixes of false traces.

6.2 Parametric Exploration

Parametric exploration is an automatic way to perform statistical model checking on a family of properties that differ by the value of a constant. The family of properties is specified in a compact way as a parametric property ϕ(x). To run a parametric exploration, follow these steps:

1. Select a 'system' file
2. Select a parametric property or edit a new one.
3. Note that the two files must be in the same project.
4. Launch a parametric exploration by clicking on the button in the toolbar
5. Set all the analysis parameters
6. Set the parametric variable parameters, that are, its name and value range
7. Start the analysis by pressing the button 'Start simulation'
8. A dialog window appears and shows the progression of the simulation
9. At the end a result tab is added to the central panel containing a summary of the analysis

6.3 The Rare-Event Engine

Importance Splitting is a technique to evaluate rare properties against stochastic systems. The condition to use it is to be able to write the property ϕ under interest in an implicative form, that is, distinguish n intermediate levels of increasing rarity, such that $\phi_i \Rightarrow \phi_{i-1}$, and $\phi = \phi_n$.

We represent these levels as a scoring function $S : \Omega \longrightarrow \mathbb{N}$, that returns, for a given trace $\omega \in \Omega$, the highest level i of satisfied intermediate property ϕ_i. Hence, a trace ω satisties the property ϕ whenever $S(\omega) = n$.

To run a rare-event analysis, follow these steps:

1. Select a 'system' file.
2. Click on the 'rare event analysis' button in the toolbar.
3. A new tab is added to the central panel, where you can specify the score function, and set rare events parameters.
4. In the Score Function panel, you can specify your score function by indicating the state formula corresponding to each level. You can also load a previously saved function. Note that score functions are checked for well-formness: (1) the score function must contain a level 1, and (2) the levels are contiguously defined, i.e., if the highest level is n then there must be exactly n defined levels with distinct score values.
5. Specify the number of maintained traces m, the confidence parameter α and the stop condition.
6. Press the 'run simulation' button to start the analysis.
7. A dialog window opens and shows the exploration progression.
8. At the end of the exploration one can find in the dialog window the following information:
   - The conditional probability for each level
   - The global estimated probability to satisfy ϕ
   - The confidence interval for the estimated probability
   - And, the execution time.

NB : Note that the rare-event engine only supports BIP2 models. It is mandatory to compile the model with the option '–gencpp-enable-marshalling' to enable storing/loading system states. You can add this option by updating the untimed BIP compilation script through the "Configure BIP compilation scripts" button in the toolbox.

7 Contact Us

$\mathcal{S}$BIP framework is an environment for BIP modeling and Statistical Model Checking developped at the laboratory Verimag, Univ. Grenoble-Alpes, France. Additional information about us can be found at http://www-verimag.imag.fr/.

This version is maintained by Braham Lotfi Mediouni and Ayoub Nouri. Please contact us at the following e-mail addresses: braham-lotfi.mediouni@univ-grenoble-alpes.fr or ayoub.nouri@univ-grenoble-alpes.fr for any information or for reporting a bug.