GUI for FIG: Visualising simulation results

Research Paper

Casper Plentinger
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
c.s.plentinger@student.utwente.nl

ABSTRACT

Statistical model checking (SMC) is used to reason about complex stochastic systems. Analytical and numerical approaches are often infeasible because of large state spaces. The Finite Improbability Generator (FIG tool) is a rare event simulator that can reason about these systems. We will be creating a GUI for FIG to visualise simulation results in real-time. Additionally, hypothesis testing will be added to the tool as an extension.

1. INTRODUCTION

1.1 Background

FIG (the Finite Improbability Generator) is a rare event simulator. Rare event simulators tackle a common issue with statistical model checking: Simulating probabilities using regular Monte Carlo techniques is infeasible for rare event models. Standard model checking methods require large sample sizes and long simulation traces to encounter rare events, accurate results cannot be reached in a reasonable amount of time. Rare event simulators like FIG use special techniques to accelerate the simulation process. There are two main techniques used for rare event simulation: Importance sampling and importance splitting. *Importance sampling* is the technique of changing the probability laws driving the system model to make the rare event happen more often. *Importance splitting* clones trajectories that seem to go towards the rare event. After applying one of these techniques, a weighting procedure will be applied to remove bias from the estimation.[8]

FIG uses importance splitting: The state space is decomposed into multiple levels, the levels are based on the probability of reaching a rare event. Ideally, the rare events are at the top level and a level is higher as the probability of reaching the rare event state grows. Thus the estimation of the rare probability is obtained as the product of the estimates of the (not so rare) conditional probabilities of moving one level up. The effectiveness of the technique depends on how well the states are grouped into levels. *Importance functions* assign a value to each state based on the likelihood of reaching the rare event. Usually, the importance function is chosen or created by an expert in the area of the system. FIG is specifically designed to automatically generate this importance function. The importance function is partially used to establish thresholds where simulation paths are split. The FIG command-line tool supports multiple importance functions:

- A flat importance function, a classical Monte Carlo simulation without the importance splitting method. Notice that this method is infeasible for rare event simulation.
- An adhoc importance function, the user can manually define an algebraic function that assigns importance to the states in the state space.
- A monolithic function, this function is automatically computed. It creates a memory vector for all states. The memory requirements can be huge, depending on the state space size.
- A compositional function, composes the global importance based on information of each module.

FIG implements the RESTART (REpetitive Simulation Trials After Reaching Thresholds) algorithm. [10] The importance of a state is defined as the chance of the process entering the rare event set after it has been in this state. Like regular importance splitting, the state space is partitioned into importance subsets. The occurrence of rare events is increased by performing several simulation retrials each time the process enters an importance subset. The restart algorithm is illustrated in Figure 1 [11] for a simulation with three importance subsets and their appropriate number of retrials: Four retrials at T_1 (thin lines), also four retrials at T_2 (dashed lines) and three retrials at T_3 (dotted lines). The main simulation trial is performed like a regular simulation. It continues until it reaches a "end of simulation" condition. Each time the main trial reaches a threshold (B_i events), the trial is paused and several retrials are made. The retrials continue till they go below their starting threshold (D_i events) or when they reach the "end of simulation" condition. When all retrials are done, the main trial continues from where it paused previously. Any trial can reach the next threshold and start the splitting process.

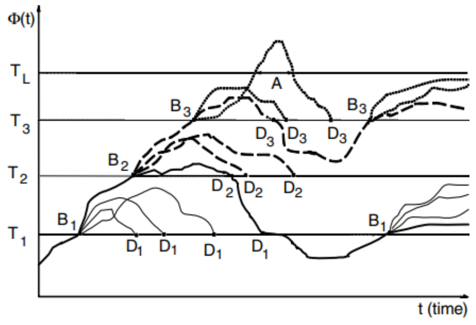


Figure 1: Simulation with RESTART [11]

Besides RESTART the FIG tool also supports a Fixed Ef-

32nd Twente Student Conference on IT January 32nd, 2020, Enschede, The Netherlands.

```
 1  // Queue capacity
 2  const int c = 8;
 3
 4  module Arrivals
 5    clk0: clock;  // External arrivals ~ Exponential(lambda)
 6    [P0!] @ clk0 -> (clk0'= exponential(1.112));
 7  endmodule
 8
 9  module Queue1
10    q1: [0..c] init 1;
11    clk1: clock;  // Queue1 processing ~ Erlang(alpha;beta1)
12    // Packet arrival (input: reacting to someone's message)
13    [P0?] q1 == 0          -> (q1'= q1+1) & (clk1'= erlang(3,3.14159265));
14    [P0?] q1  > 0 & q1 < c -> (q1'= q1+1);
15    [P0?] q1 == c          -> ;
16    // Packet processing (output: sending someone a message)
17    [P1!] q1 == 1 @ clk1 -> (q1'= q1-1);
18    [P1!] q1  > 1 @ clk1 -> (q1'= q1-1) & (clk1'= erlang(3,3.14159265));
19  endmodule
20
21  properties
22    S( q1 == c )             // steady-state
23    P( q1 > 0 U q1 == c )    // transient
24  endproperties
```

Listing 1: Queue IOSA model

fort engine. [7] [3] Unlike the RESTART method, each fixed effort simulation run performs a fixed number of retrials on each importance subset. An effort function is required to determine the number of retrials for each subset. The number of retrials that will hit the next threshold will approximately stay the same. The downside of fixed effort is that the implementation is very sensitive to the choice of effort input parameters. If the parameters are not chosen accurately, the paths will “die-out” or “explode”, leading to excessive simulation time and high memory requirements.

The FIG tool uses the IOSA input language. [6] IOSA stands for: Input/Output Stochastic Automata with Urgency. [1] Models need to be converted into IOSA for FIG to run simulations. An IOSA model is split up in modules that interact with each other by actions. The IOSA language splits actions into input and output actions. Input actions act reactively and they are controlled externally. Output actions act in a generative way. They are locally controlled by the expiration of a clock. [5]

Listing 1 shows a simple queue model in IOSA. It contains definitions of clocks and queues. The clocks sample values from a probability density function (e.g. exponential or Erlang distribution) and count down at the same rate. Clocks that reach zero can trigger events (P0 and P1 in the example). Decorators ?/! indicate an input/output action respectively. For example, line 13 indicates: When action P0 is triggered (triggered by clock 0 on line 6) and the queue (q1) is empty, increase the queue by one and set clock 1 to a sample from an Erlang distribution. When clock 1 expires and there is one message in the queue, line 17 is triggered and the queue is decreased by one and empty again. The properties block defines two queries: The steady-state query (line 22) queries the probability of observing a queue which is constantly at maximum capacity. The transient query (line 23) queries the probability of reaching the maximum capacity before the queue becomes empty. The queue starts with one packet ($q1 > 0$).

The simulations with FIG can be bounded by time or/and confidence. The command-line query for the example model in listing 1 can look like:

```
fig queue.sa --flat --stop-time 3s
```

Of course, the query is highly customisable: A different importance function, stop time or stop confidence can be used. The output from FIG contains model parsing/checking information, possible errors and the results. The results for the example model are displayed in listing 2.

It seems apparent that the IOSA modelling language is modular and flexible. However, it is not very user-friendly. One of the aims of the interface is aiding the user of FIG to write IOSA modules.

```
Time-out Interruption

   - Computed estimate: 1.11e-01
                        (349 samples)
   - Computed variance: 3.10e-05
   - 80% confidence
       - precision: 8.12e-04
       - interval: [ 1.11e-01, 1.12e-01]
   - 90% confidence
       - precision: 1.07e-03
       - interval: [ 1.11e-01, 1.12e-01]
   - 95% confidence
       - precision: 1.31e-03
       - interval: [ 1.11e-01, 1.12e-01]
   - 99% confidence
       - precision: 1.85e-03
       - interval: [ 1.10e-01, 1.12e-01]
   - Estimation time: 3.00 s

Time-out Interruption

   - Computed estimate: 1.75e-01
                        (1013440 samples)
   - Computed variance: 1.44e-01
   - 80% confidence
       - precision: 1.03e-03
       - interval: [ 1.74e-01, 1.75e-01]
   - 90% confidence
       - precision: 1.35e-03
       - interval: [ 1.74e-01, 1.75e-01]
   - 95% confidence
       - precision: 1.66e-03
       - interval: [ 1.74e-01, 1.75e-01]
   - 99% confidence
       - precision: 2.34e-03
       - interval: [ 1.73e-01, 1.76e-01]
   - Estimation time: 3.00 s
```

Listing 2: Queue IOSA model output

Apart from the graphical user interface, we will also be adding hypothesis testing to the FIG tool. There are several hypothesis tests for SMC, for instance: A sequential probability ratio test (SPRT), a Chernoff-Hoeffding test or an Azuma test. [9] Not all are applicable for rare event simulation. For a lot of hypothesis tests, the variance of the distribution must be known or estimated beforehand. This is not possible for automated rare event simulation. Therefore, the Chow-Robbins hypothesis test [4] is suitable. The Chow-Robbins test does not require knowing the variance beforehand, it can be estimated during the simulation. The Chow and Robbins theorem proofs that if one wants a confidence interval of predetermined width, one can just keep adding samples and increase N until the Central Limit Theory (CLT) indicates that this width has been reached, based on the observed variance. The Chow-Robbins test is a class-II test: It risks terminating inconclusively if the probability of the hypothesis is close to the probability of the null hypothesis. Hypothesis testing requires fewer simulation traces than calculations of confidence intervals. Only one of the bounds of the confidence interval is of importance for hypothesis testing. It is not required to reduce the width of the confidence interval, only a single bound has to be tightened for verification. [9]

1.2 Problem Statement and Objectives

As displayed in listing 2, the simulation results are not user-friendly and the confidence intervals given are hard to compare and reason about. Currently, (rare event) simulation tools rarely come with a graphical user interface. Most tools only have a command-line interface and the simulation results are given as a numerical result, often a confidence interval. Graphically displaying simulation results can be very helpful, especially if it shows results in real-time. The FIG tool lacks a graphical user interface, we will be creating this interface consisting of a simple editor and a real-time graphical result display. The main objectives of the interface will be user-friendliness. Secondly, we want to increase the productivity of the user creating the models and visualising results.

We will also be adding a completely new feature to the tool, namely hypothesis testing. This reduces the simulation time for hypothesis verification in comparison to the confidence interval calculation.

1.3 Research Question

According to the problem statement and objectives, we can define two main questions with multiple sub-questions:

1. How can we design a user-friendly graphical user interface for the FIG simulation tool?

   Sub-questions:

   (a) What are the important requirements for the interface?

   (b) What design choices can we make to improve usability?

   (c) How does the interface improve productivity?

2. How can we implement hypothesis testing for the FIG simulation tool?

   Sub-questions:

   (a) What is the best hypothesis testing algorithm to implement for FIG?

   (b) How can we formulate the query syntax for hypothesis testing?

   (c) How will we display hypothesis testing (results) in the interface?

1.4 Methodology and Approach

We will be creating the graphical user interface using the programming language Python and the Qt5 graphical interface framework. It will include an editor, result display and a panel for the FIG run configuration. The editor will be used to create and modify models such as the example in listing 1. The FIG tool will be extended such that it returns intermediate simulation results instead of one confidence interval as a result. The interface can use these intermediate results to show confidence intervals in real-time in a graph. The python subprocess module will be used to call FIG tool commands. Communication between the interface and FIG will be handled using a file I/O or standard output.

The final version of the interface is tested and assessed using a survey and hands-on experience by a user group that works with simulation tools. We let the users perform certain tasks and observe how they behave and use the interface. From the feedback collected we make important design choices and adjustments such as:

- Arranging interface elements correctly to make simulations as easy as possible.
- Displaying simulation results in a user-friendly way.
- Preventing common mistakes and complications during the simulation process.

For hypothesis testing, we will create a separate back-end in Python that does all the hypothesis calculations and instructs FIG. The simulation runs used to establish the hypothesis result will be displayed in the interface. We will implement the Chow-Robbins hypothesis test because it suits the automated simulation technique of FIG.

2. INTERFACE DESIGN

As described in the approach, the implementation of the interface was written using the Qt5 library for python. Additionally, the QtChart extensions of Qt5 is used for displaying the simulation results. The interface overview is displayed in figure 3. It shows a simple menu bar at the top with only one “File” submenu. This submenu allows the user to create, save or open a model. Additionally, the user can access the settings dialog and exit the application from this menu. Although the menu is fairly empty, it allows for new menu options in future extensions/updates.

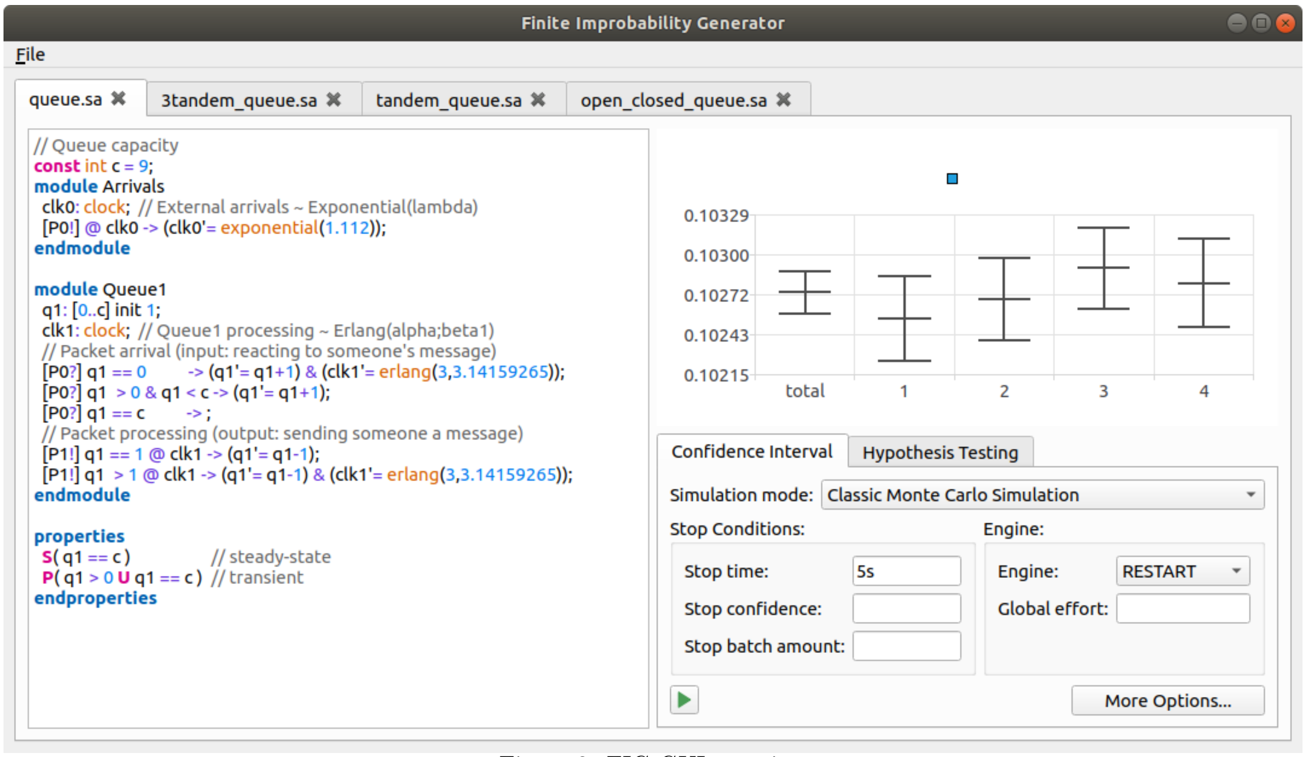
For every model that is loaded a new tab is created in the tab widget. Every model tab contains a code editor on the left and a chart view on the top-right. The options widget on the bottom-left is used for specifying simulation input parameters.

2.1 Code Editor

All the opened models will display the file contents in the code editor. The code editor supports basic text editor commands such as cut, copy, paste, undo and redo. Furthermore, the editor uses an IOSA syntax highlighter. The syntax highlighter makes the editor more user-friendly and it gives a better overview of functions, variables and code blocks.

2.2 Interactive Charts

The interface supports two types of interactive charts: One for confidence intervals and one for hypothesis testing. The chart view in the top-right switches the graphs based on the active options tab (Confidence Interval or Hypothesis Testing). When a simulation is started from



```
// Queue capacity
const int c = 9;
module Arrivals
  clk0: clock; // External arrivals ~ Exponential(lambda)
  [P0!] @ clk0 -> (clk0'= exponential(1.112));
endmodule

module Queue1
  q1: [0..c] init 1;
  clk1: clock; // Queue1 processing ~ Erlang(alpha;beta1)
  // Packet arrival (input: reacting to someone's message)
  [P0?] q1 == 0       -> (q1'= q1+1) & (clk1'= erlang(3,3.14159265));
  [P0?] q1  > 0 & q1 < c -> (q1'= q1+1);
  [P0?] q1 == c       -> ;
  // Packet processing (output: sending someone a message)
  [P1!] q1 == 1 @ clk1 -> (q1'= q1-1);
  [P1!] q1  > 1 @ clk1 -> (q1'= q1-1) & (clk1'= erlang(3,3.14159265));
endmodule

properties
  S( q1 == c )           // steady-state
  P( q1 > 0 U q1 == c )  // transient
endproperties
```

Simulation mode: Classic Monte Carlo Simulation

Stop Conditions:
Stop time: 5s
Stop confidence:
Stop batch amount:

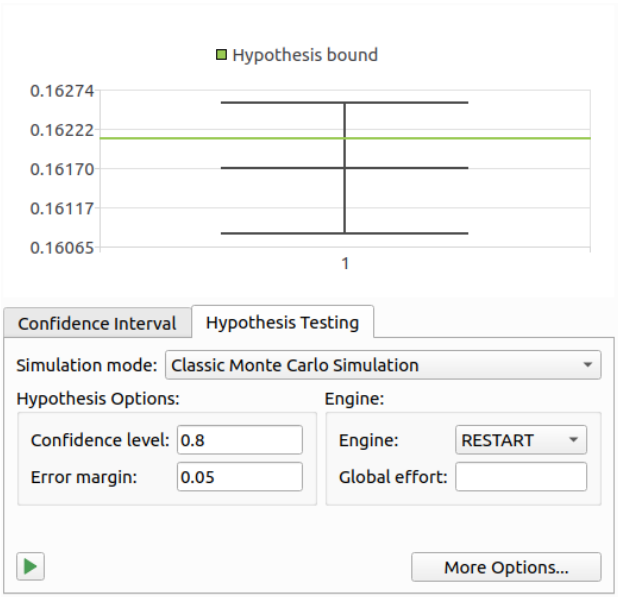
Engine:
Engine: RESTART
Global effort:

Figure 2: FIG GUI overview

the options widget, the chart view will be reset and the simulation data will show up in the chart in real-time. Every half a second the active confidence interval will be updated and the axis will be scaled appropriately. When the user hovers over a confidence interval with the mouse cursor, a tooltip will show with the mean and the bound values of the interval.

The confidence interval chart (see figure 3) always shows a “total” confidence interval representing the total of all simulation results. When a stop condition is reached, the current confidence interval will be stopped and a new interval will be added to the graph. For example, a stop time of five seconds will add a new interval to the graph every five seconds. The same holds for the stop confidence, a new interval will be started when the specified confidence level and precision are reached. The stop batch amount will create a new interval when a certain amount of simulation batches has been reached. The simulation will run indefinitely until the user stops the simulation.

The hypothesis testing chart (figure 3) shows one confidence interval and its relation to the hypothesis bound. The interval will update and move around the bound. After the stop condition has been reached, the hypothesis result will be shown in a dialog.



Simulation mode: Classic Monte Carlo Simulation

Hypothesis Options:
Confidence level: 0.8
Error margin: 0.05

Engine:
Engine: RESTART
Global effort:

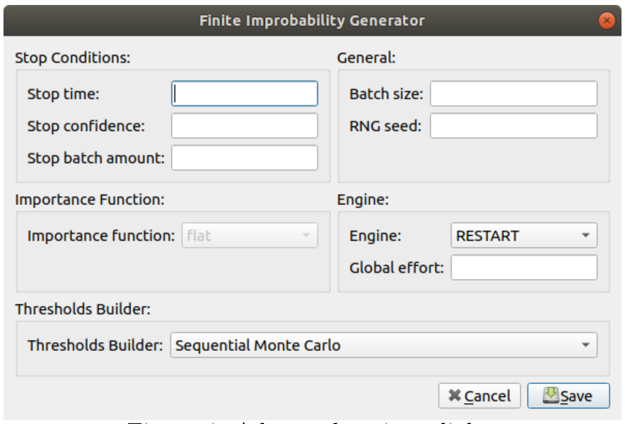
Figure 3: Hypothesis testing chart

2.3 Simulation Options

The options widget has an options tab for the confidence interval chart and the hypothesis testing chart. The most relevant and important options are displayed in the options tab. More advanced options can be accessed in a dialog using the “more options” button 4. The options in the options widget and the advanced options dialog correspond to arguments for the FIG command-line interface.

The relevant options for confidence intervals include the stop conditions and the engine. The stopping conditions influence when new confidence intervals are created in the graph. The engine can be either RESTART or fixed effort. Additionally, the global effort can be specified: A comma-separated list of splitting values that indicate how many retrials should be made on each threshold level during the simulation.

For hypothesis testing, the most relevant options are the confidence level, the error margin and the engine. When the simulation is started, the user can specify the hypothesis test in a dialog 5. The user can specify the property to test, the value of the bound and whether the probability of the property lays below or above the bound.


Finite Improbability Generator

Stop Conditions:

Stop time:

Stop confidence:

Stop batch amount:

General:

Batch size:

RNG seed:

Importance Function:

Importance function: flat

Engine:

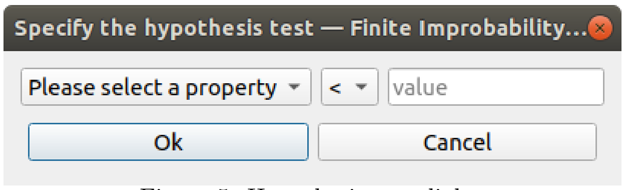
Engine: RESTART

Global effort:

Thresholds Builder:

Thresholds Builder: Sequential Monte Carlo

Figure 4: Advanced options dialog


Specify the hypothesis test — Finite Improbability...

Please select a property | < | value

Figure 5: Hypothesis test dialog

2.4 Design Choices

During the design process we made several design choices with the intent to improve the usability of the tool:

- We want the user to be able to multitask with multiple models. The most important aspects are comparing graphs and editing code of different models.
- The most important simulation options are easily accessible and displayed under the graph (inside the options widget). The simulation mode affects what the most important options are and they are displayed accordingly.
- The visualisation of data should be in real-time. This gives the user an impression of how the result changes over time.
- The code editor should have the basic features such as cut, copy and paste. Additionally, syntax highlighting makes it more organised.

One of the main goals of the project is to improve the user's productivity. We will try to achieve this by the following:

- We combine the code editor and the graph view in one interface side by side.
- Models can be modified, simulated and compared in the interface.
- Simulations results will be given in real-time instead of after all calculations.

3. DATA PROCESSING

The back-end of the interface takes the raw simulation samples from the FIG command-line standard output. These simulations samples will be processed and displayed in the interface.

The interface uses a modified version of FIG. [2] The version is adopted to give simulation samples in real-time instead of confidence intervals at the end of all calculations. Furthermore, this version of FIG supports a property index parameter (*-i* followed by an integer or list of integers). This parameter selects properties based on the index in the "properties" block in the model (shown in Listing 1 lines 21 to 24). This allows the user to select a specific property to simulate from the interface.

3.1 Confidence Intervals

The back-end uses the gathered samples to construct a confidence interval. Often, a lot of simulation samples are needed to establish reasonable confidence for rare event simulation. Therefore, doing calculations using all samples can be inefficient. To calculate the confidence intervals we use incremental computation of the mean and variance. For every new value, we will adjust the mean and variance accordingly, this prevents large sum performance problems and it lowers the significance loss that comes with summing nearly equal floating-point values.

4. HYPOTHESIS TESTING

The hypothesis testing is similar to a normal simulation, a single confidence interval is created based on all simulation samples. On every visual graph update (every half a second) the stop condition will be checked. If the stop condition is valid, we terminate the hypothesis test and return the result: Accepted, rejected or inconclusive. The result is based on the position of the confidence interval. If the confidence interval overlaps with the hypothesis bound, then the result will be inconclusive. In the other case, when the confidence interval does not overlap with the hypothesis bound, the result will be accepted or rejected depending on the user-defined direction of the hypothesis test. We implemented the Chow-Robbins hypothesis test. [4] The hypothesis test is ended as soon as the half-width of the interval has reached the user-defined error margin.

5. USABILITY TESTS

We performed usability tests for the complete version of the interface. To be specific, a cognitive walkthrough with users that are familiar with statistical model checking and simulation. Preferably, we want users that work with similar simulation tools. We found six people that fitted the target group and conducted the usability test. We asked six basic tasks for the user to perform. While the user performs these tasks we noted down problems and feedback. One of the six users performed the test remotely and submitted the test results. The tasks and a feedback summary are shown in appendix A. After performing all tasks we asked the user for overall/final feedback and any other encountered problems during the test.

We will go over each task of the usability test (Appendix A, Figure 6) and discuss the results:

1. All participants were able to open the example model. The "Open Model" menu option is easy to find under the "File" submenu because there are no other submenus. One user noticed that if one or more models are opened in the interface, newly opened models are shown in the background and the model tab is not made active.
2. All users were able to increase the queue capacity value in the example model. Three out of five users saved the model using the Ctrl+S shortcut without being asked to. The other users managed to save the model using the "Save Model" option in the "File" submenu after being asked to save the model.

3. All the users found the option input field for the stop time condition. Two users were able to enter the stop time correctly (the value "10s"). Two users expressed their confusion about how the time units should be specified before going forward. One user expected the stop time to be unitless and entered the value "10". All users found the "start simulation" button at the bottom of the options widget. Two users mentioned that the start button was small and should stand out more. One user suggested displaying the elapsed simulation time.

4. All users were able to stop the simulation. However, most users were surprised that the simulations continue indefinitely and can only be interrupted manually by the user. One user suggested stopping the simulation after a user-specified number of confidence intervals. All users were able to switch the simulation mode. Two users questioned how the simulation mode influences the simulation, this is unclear from the interface. Two users forgot to enter the stop time of ten seconds again. We can consider remembering the stop conditions when the simulation mode changes. All users were able to determine the mean of the first confidence interval by hovering over the interval in the graph. However, two users pointed out that reading off the value using the y-axis is hard because of the not-rounded axis values.

5. The users were able to switch to the hypothesis testing options widget without any difficulty. Only three users were able to enter the confidence level and error margin correctly in a decimal format (confidence level "0.8" and error margin "0.05"). The other users entered the percentages in a different format and encountered errors. All users were able to enter the correction options in the hypothesis test dialog (Figure 5) and run the test. Three users had questions about how the hypothesis works: When the test is stopped and how it determines the result.

6. All users were able to switch to rare event simulation. One user was unable to perform the second hypothesis test at all. Two users forgot to enter the confidence level and error margin again. All users pressed the "more options" button to find the threshold builder option. One user was confused about what options in the dialog were required.

From the usability tests, we can determine changes that will improve the user's productivity. There are multiple problems that hold back user productivity. For example, there is no information for simulation options and the graphs are reset after each simulation. To increase productivity we want to keep features and data easily accessible to the user. That is why we allow multiple models to be opened in the interface, the same should be done for graphs.

From the gathered results we can come up with the most important points of improvement:

- Simulation options need extra information. It is unclear what the options mean. Additionally, there is no distinction between required and optional options. We can implement tooltips for each simulation option to give a help message. Also, the required options can be marked using an asterisk symbol.

- The axis of the graphs should only be aligned and adjusted when the confidence intervals fall outside of the plotting area. The axis should also have properly rounded values, this makes reading off graphs a lot easier. A "realign graph" button can be considered for the user to manually adjust axis. Additionally, the graphs should not be cleared when a simulation is started. It would also be useful to maximise the graph view during simulation or in other words: Hiding the code editor based on the user's preference. Instead, the old graph should stay in memory and the user should be able to save a picture of the graph to disk.

- The input values for option fields can be sanitised better. Error messages are vague and are often not helpful for the user.

- The hypothesis test should be explained better: What the confidence level and error margin mean, how the Chow-Robbins test is used and what the test result means.

6. CONCLUSION

By defining several important design choices, and using usability tests to gather feedback, we have created an interface that visualises the simulation data in a modern and organised way. Although the interface has issues and the usability can be improved, we have stated the usability problems we found and gave suggestions to improve it. The primary goals of the research have been reached: Displaying simulation results in real-time, and implementing hypothesis testing for the FIG tool. The usability tests showed that users familiar with statistical model checking are able to edit simulation models, perform simulations and read off the result data. Advanced simulations require more knowledge of the tool than the interface currently provides, this is one of the most crucial points of improvement.

7. REFERENCES

[1] Finite improbability generator (fig tool). https://git.snt.utwente.nl/buddece/fig.

[2] Modified version of the finite improbability generator (fig tool). https://git.snt.utwente.nl/s1864335/fig/tree/fig$_{a}s_{l}ibrary$.

[3] C. E. Budde, P. R. D'Argenio, and A. Hartmanns. Better automated importance splitting for transient rare events. In *Dependable Software Engineering. Theories, Tools, and Applications - Third International Symposium, SETTA 2017, Changsha, China, October 23-25, 2017, Proceedings*, pages 42–58, 2017.

[4] Y. S. Chow and H. Robbins. On the asymptotic theory of fixed-width sequential confidence intervals for the mean. *The Annals of Mathematical Statistics*, 36(2):457–462, 1965.

[5] P. R. D'Argenio, M. D. Lee, and R. E. Monti. Input/output stochastic automata - compositionality and determinism. In *Formal Modeling and Analysis of Timed Systems*, pages 53–68, 2016.

[6] P. R. D'Argenio and R. E. Monti. Input/output stochastic automata with urgency: Confluence and weak determinism. In *Theoretical Aspects of Computing - ICTAC 2018*, pages 132–152, 2018.

[7] M. J. J. Garvels and D. P. Kroese. A comparison of RESTART implementations. In *Proceedings of the 30th conference on Winter simulation, WSC 1998,*

Washington DC, USA, December 13-16, 1998, pages 601–608, 1998.

[8] P. L'Ecuyer, M. Mandjes, and B. Tuffin. Importance sampling in rare event simulation. In *Rare Event Simulation using Monte Carlo Methods* [8], pages 17–38.

[9] D. Reijsbergen, P. de Boer, and W. R. W. Scheinhardt. Hypothesis testing for rare-event simulation: Limitations and possibilities. In *Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques*, pages 16–26, 2016.

[10] M. Villén-Altamirano and J. Villén-Altamirano. Restart: a method for accelerating rare event simulations. In *Queueing, Performance and Control in ATM (ITC13)*, pages 71–76, 1991.

[11] M. Villén-Altamirano and J. Villén-Altamirano. The rare event simulation method RESTART: efficiency analysis and guidelines for its application. In *Network Performance Engineering*, pages 509–547. 2011.

8. APPENDIX A: USABILITY TESTS

Nr.	Task
1	Can you open the queue model file?
2	Are you able to increase the queue capacity in the model by one and save the changes you just made?
3	Could you show a confidence interval with a stop time of 10 seconds?
4	The next step is to stop the simulation and to rerun it as a rare event simulation with the same stop time. Are you able to determine the mean of the first confidence interval?
5	Now we want to run a hypothesis test with a confidence level of 80% and an error margin of 5%. We want to test if the probability of the last property in the model is above a value of 0.096.
6	Can you run the same hypothesis test using rare event simulation and a different threshold builder?

Figure 6: Usability test tasks

Task nr.	Feedback and problems
1	Saving a newly created model will give an error (path not defined).
2	Editor is very basic and lacks advanced features such as auto-completion and error highlighting. Saving the model resets the cursor position to the first character of the editor.
3	The simulation options have no explanation or help message, users are confused about input syntax. Users often forget to define time units for the stop time and enter "10" instead of "10s". Decimal stop time values are not supported. Users are unsure which simulation options are required and which are optional.
4	Users are unsure how the "simulation mode" option affects the simulations. Changing simulation mode clears the stop condition option fields. All users were able to determine the mean of the first confidence interval. The axis are hard to read off because of not-rounded axis values. Axis are realigned on every graph update, this makes the confidence intervals jump around. The elapsed time for a running simulation is not shown. Using the ad-hoc importance function without a formula gives a generic parse error. Graphs are reset (not saved) when a new simulation is restarted. Graphs can not be saved to file.
5	The syntax for percentages is not clearly defined. Only decimal values are currently supported. The meaning of confidence level and error margin are not clearly defined. The results of the hypothesis tests do not show the simulation time.
6	Switching simulation mode clears field values. Running a hypothesis test without the required input fields does not give an error at all.

Figure 7: Feedback from the usability tests.