

Developing Data-Intensive Cloud
Applications with Iterative Quality
Enhancements



DICE simulation tools - Intermediate version

Deliverable 3.3

Deliverable:	D3.3
Title:	DICE simulation tools - Intermediate version
Editor(s):	Diego Perez (ZAR)
Contributor(s):	Simona Bernardi (ZAR), José Merseguer (ZAR), José Ignacio Requeno (ZAR)
Reviewers:	Ismael Torres (PRO), Youssef Ridene (NETF)
Type (R/P/DEC):	Report
Version:	1.0
Date:	31-January-2017
Status:	Final version
Dissemination level:	Public
Download page:	http://www.dice-h2020.eu/deliverables/
Copyright:	Copyright © 2017, DICE consortium – All rights reserved



The DICE project (February 2015-January 2018) has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 644869

Executive summary

This document presents the intermediate results of the development of the *Simulation tool*. It details the advances of this tool with respect to its initial version reported at M12. Achievements in this intermediate version are both in the functionality offered by the tool and in its graphical interaction with the user. This deliverable is related to D1.2 (*Requirement specification*) as it evaluates the level of accomplishment of requirements achieved by this intermediate version of the *Simulation tool*.

All the artifacts presented in this document are publicly available in the so-called *DICE-Simulation Repository* [10], whose structure and components are described in the Appendix A of this document.

Glossary

DAM	Dependability Analysis and Modeling
DIA	Data-Intensive Application
DICE	Data-Intensive Cloud Applications with iterative quality enhancements
DPIM	DICE Platform Independent Model
DTSM	DICE Technology Specific Model
GUI	Graphical User Interface
IDE	Integrated Development Environment
M2M	Model-to-model Transformation
M2T	Model-to-text Transformation
MARTE	Modeling and Analysis of Real-time Embedded Systems
MDE	Model-Driven Engineering
OSGi	Open Services Gateway initiative
PNML	Petri Net Markup Language
QVT	Meta Object Facility (MOF) 2.0 Query/View/Transformation Standard
QVTc	QVT Core language
QVTo	QVT Operational Mappings language
QVTr	QVT Relations language
UML	Unified Modelling Language

Contents

Executive Summary	3
Glossary	4
Table of Contents	5
List of Figures	6
List of Tables	6
1 Introduction and Context	7
1.1 Structure of the Document	7
2 Summary of achievements in the initial version of the Simulation Tool	9
2.1 Requirements coverage summary	9
2.2 Summary of functionality and GUI of the initial version of the <i>Simulation tool</i>	10
3 Updates in functionality	13
3.1 Enhancements on the models that can be simulated	13
3.2 Enhancements on the quality properties that can be simulated	13
3.3 Enhancements in the simulation characteristics	14
3.4 Enhancements in the reporting of simulation results	16
4 Updates in the Graphical User Interface	18
4.1 Enhancements on the GUI to configure a simulation	18
4.2 Enhancements on the GUI to visualize the simulation results	20
5 Conclusion	24
5.1 Further Work	26
References	27
Appendix A. The DICE-Simulation Repository	28

List of Figures

1	Sequence diagram depicting the interactions between the components of the <i>Simulation Tool</i>	10
2	Create a new <i>Simulation launch configuration</i> from a workspace model.	11
3	A <i>Simulation launch configuration</i> with the simulation parameters already filled.	12
4	<i>Debug</i> perspective of the <i>Simulation tool</i> , showing the simulation Debug view on top and the simulation console on the bottom.	12
5	Sequence diagram highlighting the functionality enhancement with respect to the <i>models that can be simulated</i>	13
6	Activity diagram depicting a Storm execution scenario that can be now simulated by the <i>Simulation tool</i>	14
7	Sequence diagram highlighting the functionality enhancement with respect to the <i>quality properties that can be simulated</i>	15
8	Sequence diagram highlighting the functionality enhancement with respect to the <i>simulation characteristics</i>	16
9	Sequence diagram highlighting the functionality enhancement with respect to the <i>reporting of simulation results</i>	17
10	Simulation result of DIA throughput	17
11	A <i>Simulation launch configuration</i> window showing the Main tab.	18
12	A <i>Simulation launch configuration</i> window showing the Filters tab.	19
13	A <i>Simulation launch configuration</i> window showing the Parameters tab.	20
14	A <i>Simulation launch configuration</i> windows defining <i>what-if</i> analysis over rate variable and response time and throughput as quality metrics of interests.	21
15	View of the <i>Invocations Registry</i> with the results of a <i>what-if</i> analysis composed of nine concrete simulations	21
16	New option <i>Plot Results</i> within the <i>Invocations Registry</i> view	21
17	Window to choose the variable to use as x-axis in the plot.	22
18	Window to choose the variable to use as y-axis in the plot.	22
19	Window to choose the name and path where the plot will be saved.	23
20	Plot showing <i>what-if</i> analysis results with the correlation of how the arrival rate affects the response time.	23

List of Tables

1	Status of the <i>Simulation tool</i> at M12. Data brought from Deliverable D3.2 [6].	9
3	Level of compliance with requirements of the intermediate version of the <i>Simulation tool</i>	24

1 Introduction and Context

The goal of the DICE project is to define a quality-driven framework for developing data-intensive applications (DIA) that leverage Big Data technologies hosted in private or public clouds. DICE offers a novel profile and tools for data-aware quality-driven development. In particular, the goal of WP3 of the project is to develop a quality analysis tool-chain that will be used to guide the early design stages of the data intensive application and guide quality evolution once operational data becomes available. Therefore, the main outputs of tasks in WP3 are tools for simulation-based reliability and efficiency assessment, for formal verification of safety properties related to the sequence of events and states that the application undergoes, and numerical optimization techniques for searching the optimal architecture designs. Concretely, Task T3.2 in WP3 is in charge of developing the *DICE Simulation tool*, a simulation-based tool for reliability and efficiency assessment of DIA.

This deliverable describes the intermediate version, at M24 of the project, of the *DICE Simulation tool*. The initial version of this *Simulation tool* was already reported, at M12, in deliverable D3.2 [6].

Simulations carried out by the *Simulation tool* are model-based simulations. For performing its model-based simulation task, the tool takes as input UML models annotated with the DICE profile developed in Task T2.2. Then, it uses Model-to-model (M2M) transformations, that transform the DIA execution scenarios represented in these profiled UML models into Petri net models, evaluates these Petri nets, and finally uses the results of the Petri net evaluation to obtain the expected performance and reliability values in the domain of DIA software models.

Compared with the initial version of the tool, released at M12, the current intermediate version has enhanced both its functionality and its interaction with the user. Regarding the functionality enhancements, the *Simulation tool* has been extended with new simulation capabilities and now offers possibilities to simulate DIA models at DTSM level, simulate more quality properties of the DIA – such as reliability related properties –, configure a simulation with more configuration parameters – e.g., the specification of a *timeout* parameter for finishing the simulation of a model–, as well as a better management of the simulation results. Regarding the enhancements in the interaction of the user with the tool, the Graphical User Interface (GUI) of the tool has been extended and improved. For instance, the GUI extension allows the user to graphically configure all the possible parameters of the simulation.

The implementation of the previously mentioned M2M transformations used by the *Simulation tool* is also one of the objectives of Task T3.2. This objective is achieved in close interaction with WP2, since Tasks T2.1 and T2.2 of WP2 define the languages and notations that are used to express the input design models. Therefore, new M2M transformations are being defined and implemented in Task T3.2 whilst the tool increments its simulation capabilities. The tool is integrated in DICE IDE, and it is published as an open source software that can be downloaded from the *DICE-Simulation* repository [10].

1.1 Structure of the Document

The structure of this deliverable is as follows:

- Section 2 presents a summary of the achievements already included in the initial version of the *Simulation tool*, the fulfillment of requirements of the initial version, and introduces a new requirement to satisfy in Task T3.2.
- Section 3 presents updates of the intermediate version of the tool with respect to its offered functionality.
- Section 4 presents updates of the intermediate version of the tool with respect to its GUI.

- Section 5 Presents the updates on the fulfillment of requirements by the intermediate version of the *Simulation tool* and concludes the deliverable.

2 Summary of achievements in the initial version of the Simulation Tool

This section provides an overview of the initial version of the *Simulation tool* already reported in D3.2 on M12, and a summary of the requirement accomplishment of such initial version.

2.1 Requirements coverage summary














Deliverable D1.2 [3, 4], released at M6, presented the requirements analysis for the DICE project. The outcome of the analysis was a consolidated list of requirements and the list of use cases that define the project's goals that guide the DICE technical activities. From this requirement analysis, Deliverable D3.2 [6] gathered the requirements on Task T3.2 and reported the level of accomplishment of the initial version of the *Simulation tool* of each of these requirements. The schematic view of their **ID**, **Title** and **Priority** is provided in Table 1. It also shows, in column **Level of fulfillment**, their accomplishment in the initial version of the tool (this table has been brought from D3.2). The meaning of the labels used in column **Level of fulfillment** is the following: (i) **X** (unsupported: the requirement is not fulfilled by the current prototype); (ii)  (partially-low supported: a few of the aspects of the requirement are fulfilled by the prototype); (iii)  (partially-high supported: most of the aspects of the requirement are fulfilled by the prototype); and (iv)  (supported: the requirement is fulfilled by the prototype and a solution for end-users is provided).

Table 1: Status of the *Simulation tool* at M12. Data brought from Deliverable D3.2 [6].

Requirement	Title	Priority	Level of fulfillment
R3.1	M2M Transformation	MUST	
R3.2	Taking into account relevant annotations	MUST	
R3.4	Simulation solvers	MUST	
R3.5	Simulation of hosted big data services	MUST	X
R3.6	Transparency of underlying tools	MUST	
R3.10	SLA specification and compliance	MUST	
R3.13	White/black box transparency	MUST	X
R3IDE.1	Metric selection	MUST	
R3IDE.4	Loading the annotated UML model	MUST	
R3.3	Transformation rules	COULD	
R3.14	Ranged or extended what if analysis	COULD	
R3IDE.2	Timeout specification	SHOULD	
R3IDE.3	Usability	COULD	X

At M16, it was released an updated version [7] of the requirement specification deliverable D1.2 [3, 4]. Updates in DICE requirements in [7] that affect the work to carry out in Task T3.2 and in the *Simulation tool* are the following:

- R3.5 and 3.12 have been deprecated.
- A new requirement R3IDE.7 has been created.

The rationale for deprecating R3.5 “Simulation of hosted big data services” is that the *Simulation tool* simulates the behavior of DIA, but it does not simulate the environment of big data services where the DIA is hosted. In turn, the rationale for deprecating R3.12 “Modeling abstraction level” is that it is a requirement that the *Simulation tool* satisfies by default because this modeling abstraction level is an intrinsic characteristic of the development process used in DICE.

A new requirement R3.IDE7 has been created, with name “Output results of simulation in user-friendly format”. The rationale for creating this new requirement is that, during the accomplishment of R3.14 “Ranged or extended *what-if* analysis”, this *what-if* analysis produced several quality results, and

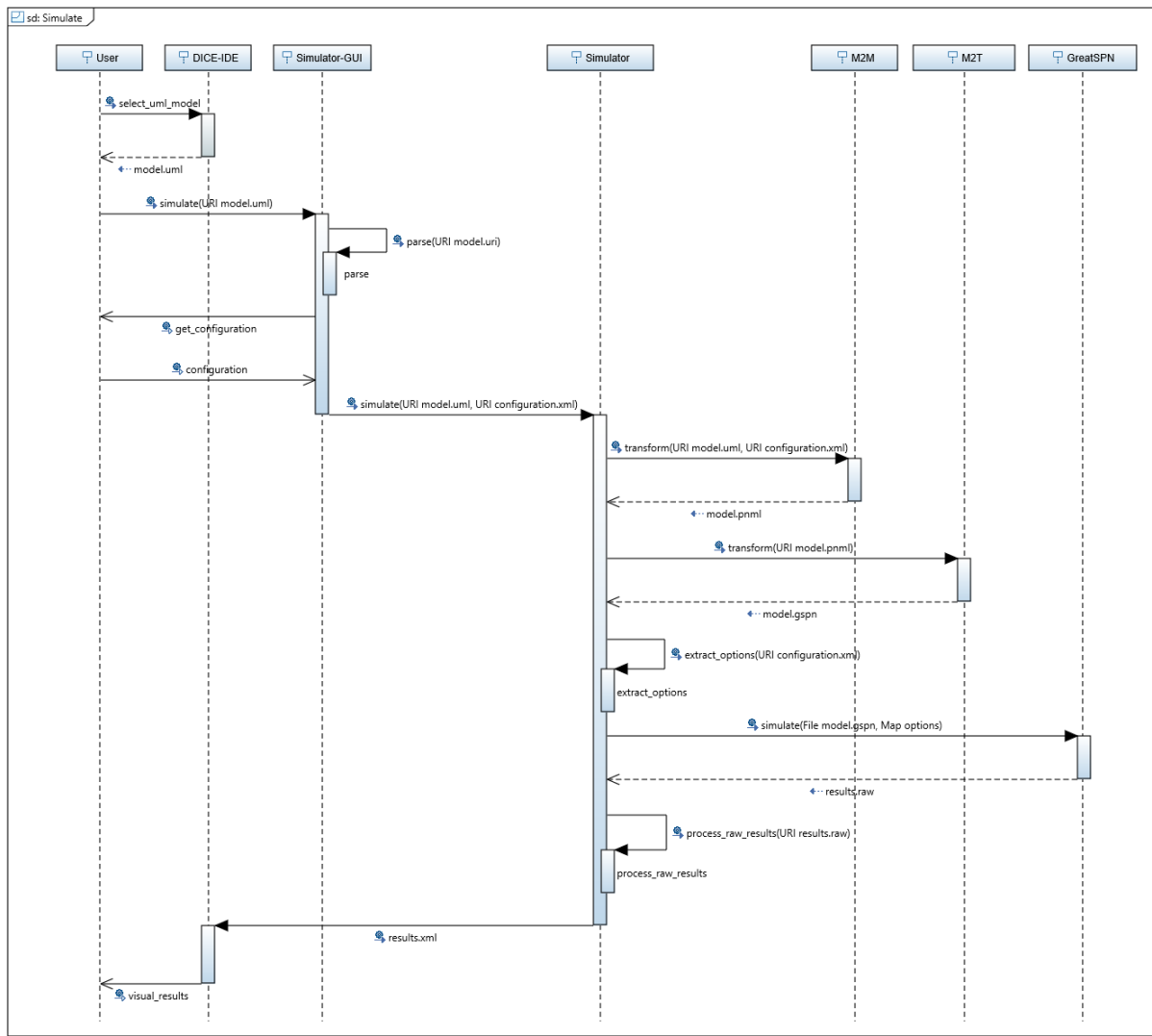


Figure 1: Sequence diagram depicting the interactions between the components of the *Simulation Tool*

developers and demonstrators realized that it was not easy to envision the big picture of the behavior of a DIA watching one at a time the multiple results provided by the *what-if* analysis. A solution passed through adding a new requirement that offers the user the possibility to automatically depict graphically the results of a *what-if* analysis. More details regarding the updates of requirements can be found in Deliverable D1.4 [8], which is submitted concurrently to this document.

2.2 Summary of functionality and GUI of the initial version of the *Simulation tool*

The behavior of the initial version of the tool and the interactions of their components is depicted in Figure 1 (brought from D3.2). It shows that the user interacts with the tool through a `simulate` of a certain UML model and through the `configuration` interaction that allows to configure the properties and possibilities of a simulation before launching it. Once the *Simulator GUI* component receives the model to simulate and the configuration values, it invokes the *Simulator* which in turn orchestrates the necessary model transformations to build an analyzable Petri net model (i.e., a sequence of model-to-model *M2M* and model-to-text *M2T* transformations), invokes *GreatSPN* Petri net simulator, and process the raw Petri net results offered by *GreatSPN* to create a set of results that are meaningful in the domain of the UML model.

The GUI of the initial version of the tool comprised:

- A shortcut menu option to run the *DICE simulation* when a UML model was selected in the IDE. Figure 2 depicts such option.
- A configuration window that allows to choose among the different types of results that the *Simulation tool* can offer for a concrete UML model and also to give actual values to the parameters left as variables in the profiled UML model. Figure 3 depicts such configuration window.
- A debug view to show the information an execution of a model simulation, e.g., identifier, state, exit value, etc. Uppermost part in Figure 4 depicts this view.
- A console view that shows messages dumped by GreatSPN, which, for instance, allows to monitor the accuracy already achieved in a simulation process while the process is running. This console view is shown in the lowest part in Figure 4.

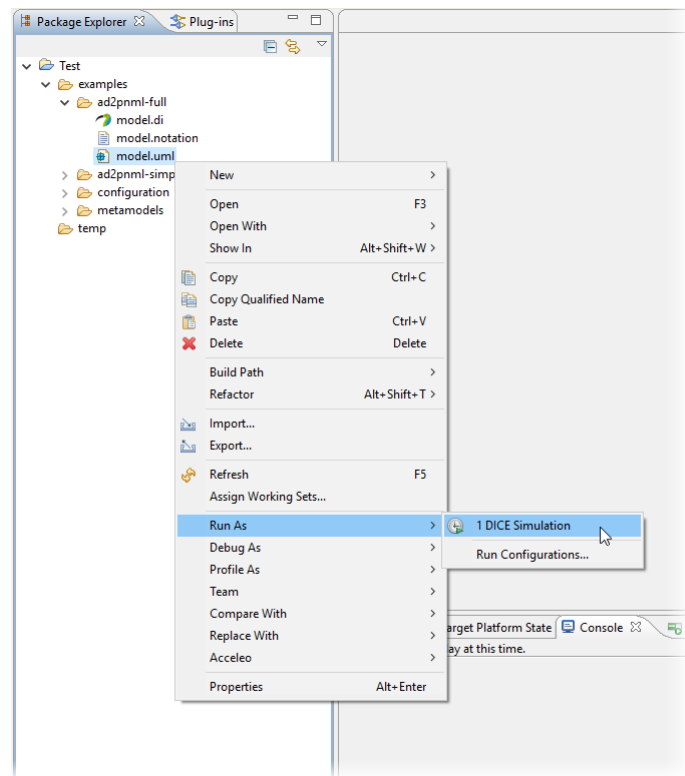


Figure 2: Create a new *Simulation launch configuration* from a workspace model.

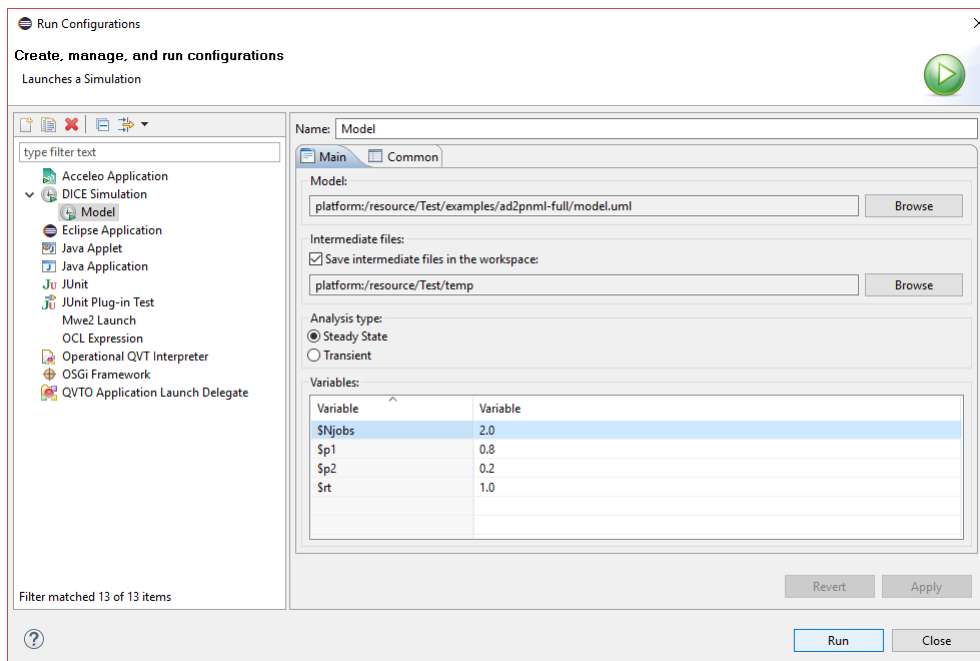


Figure 3: A *Simulation launch configuration* with the simulation parameters already filled.

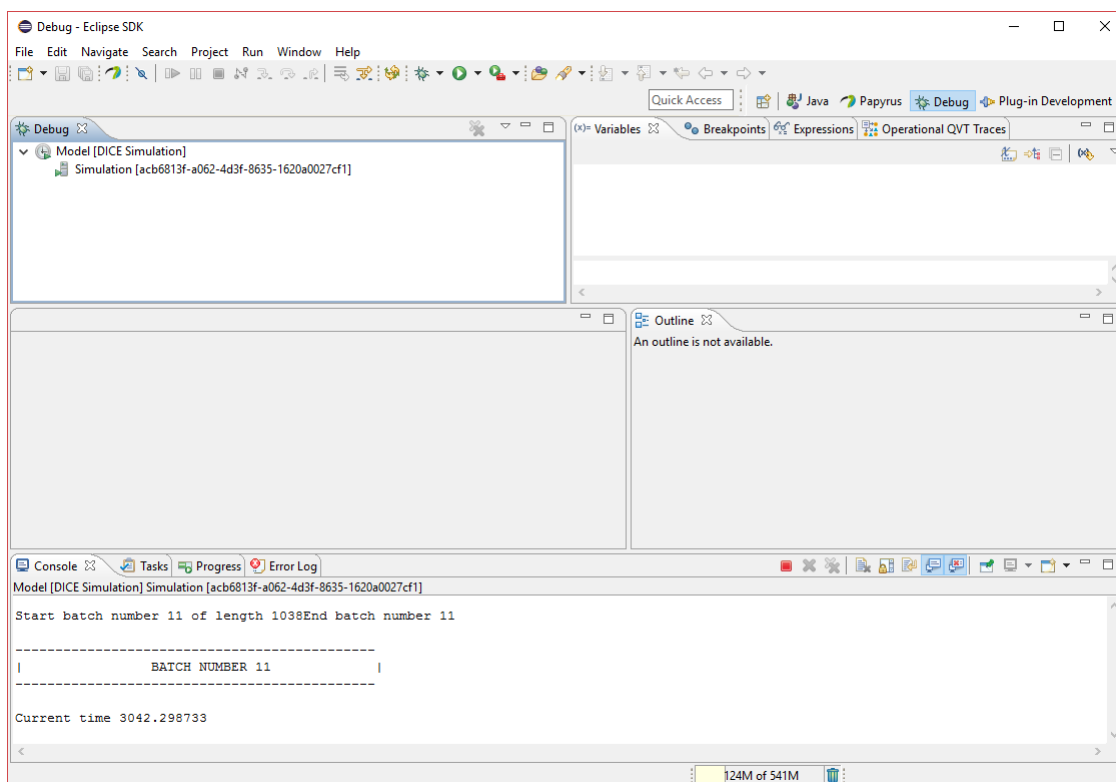


Figure 4: *Debug* perspective of the *Simulation tool*, showing the simulation Debug view on top and the simulation console on the bottom.

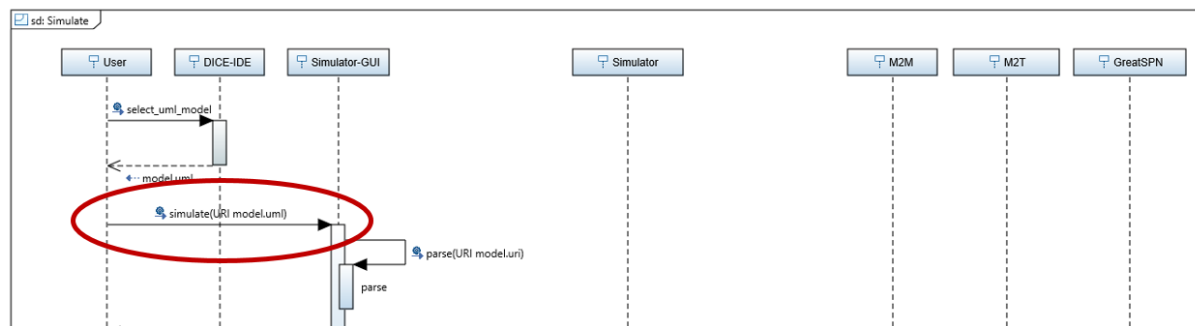


Figure 5: Sequence diagram highlighting the functionality enhancement with respect to the *models that can be simulated*

3 Updates in functionality

This section provides the enhancements in the functionality offered by the current intermediate version of the *Simulation tool*. These enhancements can be classified in four main blocks: enhancements on the models that can be simulated, enhancements on the quality properties that can be simulated, enhancements in the simulation characteristics and enhancements in the reporting of simulation results. Next subsections explain the enhancements in each of these blocks.

3.1 Enhancements on the models that can be simulated

The intermediate version of the *Simulation tool* accepts both Apache Storm and Apache Hadoop models at DTSM level; i.e., UML models that are annotated with the DICE DTSM::Storm or DTSM::Hadoop profiles, respectively. Figure 5 highlights the corresponding part of the Simulation tool design where this enhancement is perceived by the user with respect to the initial version of the tool depicted in Figure 1.

With this functionality enhancement, the user perceives that s/he can request for performance simulation a broader set of types of UML models. For instance, the user can now pass to the *Simulation tool* UML diagrams of Storm models as depicted in Figure 6 and request for its expected ‘Response time’, ‘Throughput’ or resource ‘Utilization’. However, the availability of this enhancement to the user has not only required new implementations on the receptor of the *simulate* message (i.e., the Simulator-GUI component in Figure 1) but it has required new implementations in several components of the *Simulation tool*. Concretely, it has required:

- to extend the implementation of the M2M component in order to be able to create Petri net models of the Apache Hadoop and Storm UML models,
- the enhancement of M2T transformations that take into account the new characteristics of the Petri net (e.g., deal with arcs with weight for the Storm case and deal with Colored Petri nets for the Hadoop case),
- new methods to process the raw results provided by GreatSPN in order to compute the metrics of interest within a Storm or Hadoop software topology.

This enhancement increments the satisfaction level of requirements R3.1, R3.2, and R3IDE.4.

3.2 Enhancements on the quality properties that can be simulated

The intermediate version of the tool includes the possibility to study reliability properties of a DIA. While the initial version of the tool allowed to simulate only performance metrics, the current version has been enhanced to allow to simulate also the expected reliability of a DIA in terms of the probability of failure of an execution. At present, it is possible to study the reliability of DIA UML models at DPIM level.

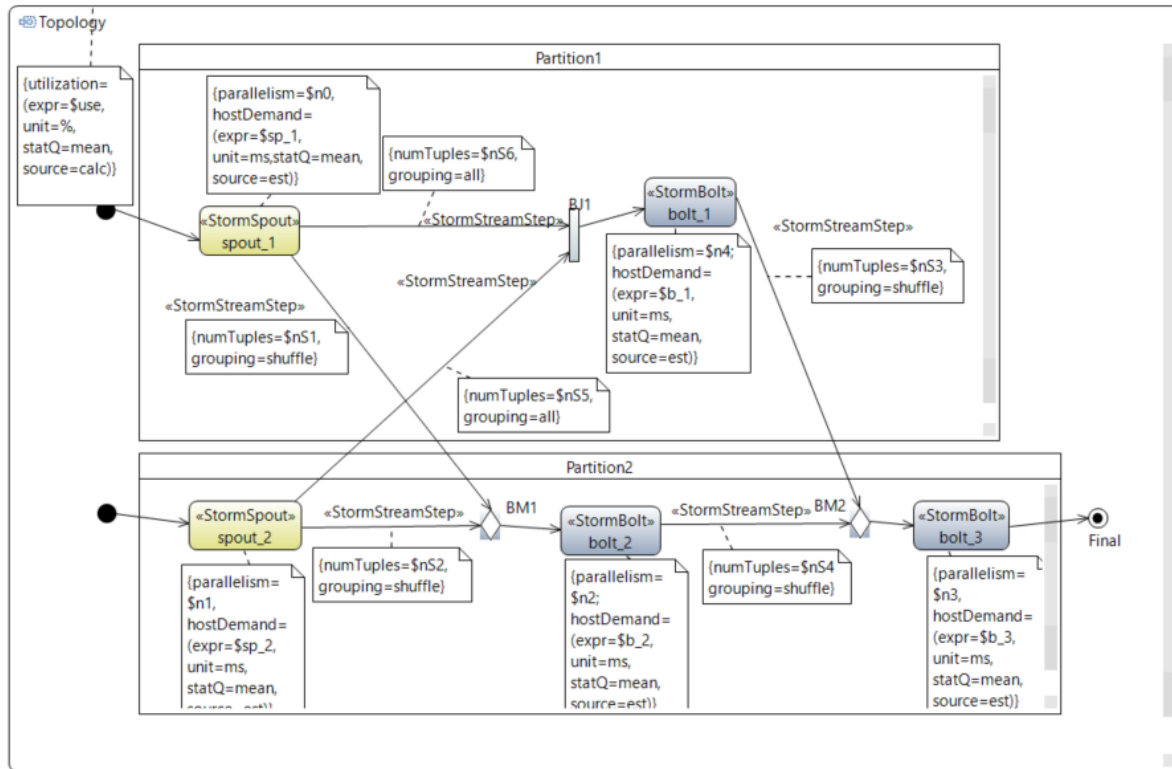


Figure 6: Activity diagram depicting a Storm execution scenario that can be now simulated by the *Simulation tool*

With this functional enhancement, the user perceives that s/he can request for simulation a broader set of quality properties of a DIA design. The parts along the interaction with the *Simulation tool* where the user perceives this enhancement are depicted in Figure 7. Regarding the first *simulate* message, the user can now ask for simulation of models that contain reliability annotations. In turn, regarding *configuration* message, the user can configure the simulation by choosing to execute either a performance or a reliability evaluation of the profiled UML DIA design.

In order to offer this new functional enhancement to the user, it has been required to work on the following elements of the *Simulation tool*:

- M2M transformations. The set of M2M transformations has been extended. In an analogous way as MARTE profile is used in the modeled scenarios for specifying the information needed for performance evaluation, stereotypes of DAM profile are now used for specifying the information needed for a model-based reliability evaluation. Therefore, a new M2M transformation has been implemented for transforming the UML model profiled with DICE and DAM into a Petri net whose structure represents the failure of activities along the DIA workflow.
- M2T transformations. M2T transformation to GreatSPN file format has been extended to support the creation of Petri nets in GreatSPN language that contain immediate transitions with priority.
- Calculators of quality metrics. A new method that processes the raw results provided by GreatSPN engine, computes them and produces a value in the domain of the reliability metric.

This functional enhancement increments the satisfaction level of requirements R3IDE.1, R3.1, R3.2.

3.3 Enhancements in the simulation characteristics

The intermediate version of the *Simulation tool* allows to simulate new characteristics of the UML models. Concretely, it is possible:

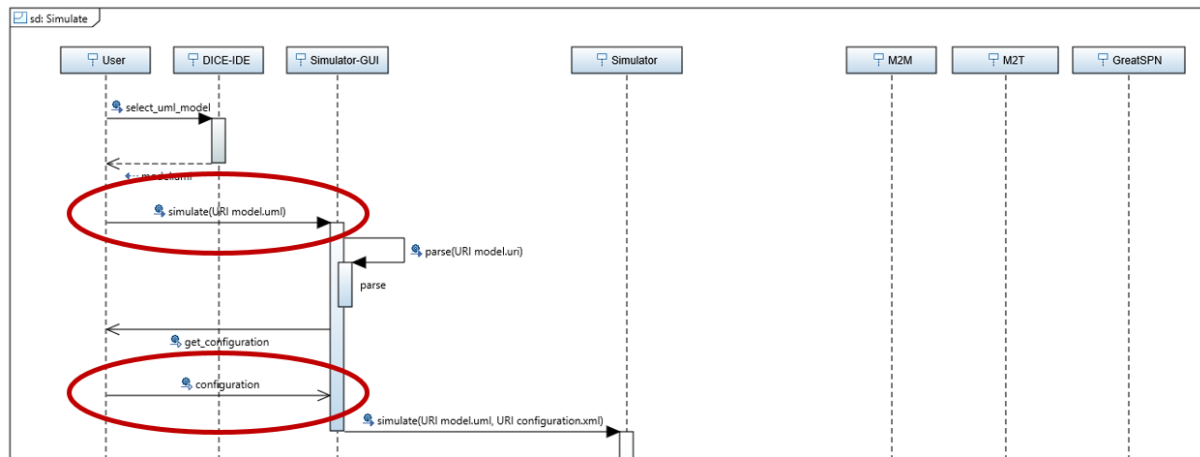


Figure 7: Sequence diagram highlighting the functionality enhancement with respect to the *quality properties that can be simulated*

- To simulate UML models that define several scenarios. In the initial version of the tool, only UML models with one scenario were allowed. However, DIAs usually have different execution scenarios, and the engineer wants to design them in a single project. Moreover, the designer may want to keep in a single project all the DIA models and execution scenarios, i.e., at DPIM, DTSM and DDSM levels. Therefore, the intermediate version of the tool is able to read all the scenarios defined in the project, to list all of them, and to allow the user to select the scenario of interest in each moment. This functional enhancement increments the satisfaction level of requirements R3IDE.1 and R3IDE.4.
- To simulate *what-if* scenarios. In the initial version of the tool, the engineer could only set a single value for each variable in the scenario. In the current intermediate version, the engineer can specify a list of values for each variable, and the tool will simulate each possible combination of the values of variables. The high-level design of the simulator core in the initial version of the tool already took into account the necessity include this functionality in the future. Therefore, this enhancement has not required a deep refactoring but only a new management of the series of invocations to the GreatSPN simulation engine and the logical models to store the results received from a set of different simulations. To include this capability, the intermediate version of the tool has extended the *Simulator* component both to execute a series of simulations and to generate *what-if* execution results by gathering the different results of each simulation, and the *Simulator-GUI* component to identify the different simulations required. New graphical guides provided by *Simulator-GUI* are detailed in next section. This functional enhancement increments the satisfaction level of requirement R3.14.
- To choose the quality metric to compute. In the intermediate version of the tool, the user may define several performance and reliability metrics of interest in a scenario. However, s/he may not be interested in evaluating all metrics in a concrete simulation. Therefore, to avoid the user continuously defining and removing quality metrics from the UML models depending on the concrete concern of interest in each simulation, the intermediate version of the tool provides a list with all defined metrics. Among this set of defined metrics, the user s/he can choose the subset of metrics to compute in a concrete simulation, and the tool will ignore the rest of the metrics defined in the model. This functional enhancement increments the satisfaction level of requirement R3IDE.1
- To specify a Timeout for the duration of the simulation. Model-based simulations can last for very long periods, depending on the characteristics of the models. In the intermediate version of the tool, the user can specify the maximum amount of time s/he can afford waiting for the simulation results. After this amount of time, if the simulation has not finished, it gets the last partial results

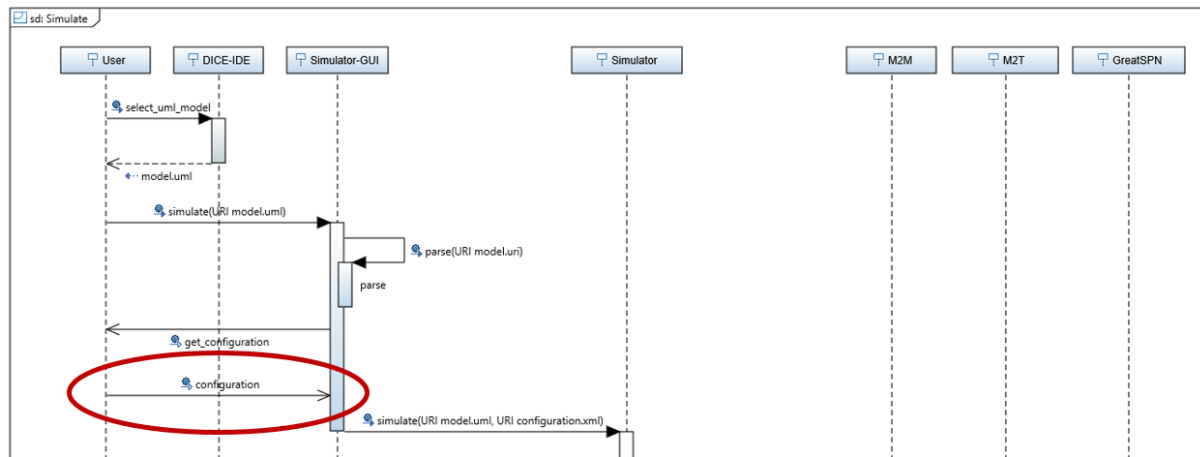


Figure 8: Sequence diagram highlighting the functionality enhancement with respect to the *simulation characteristics*

obtained and passes them to the user, together with the accuracy level achieved by the simulation. Moreover, in the results, a simulation that has finished due to timeout expiration is also graphically marked, in order to easily alert the user about the trustworthiness of the obtained results. This functional enhancement increments the satisfaction level of requirement R3IDE.2.

Figure 8 highlights the corresponding part in the *Simulation tool* design where these enhancements are perceived by the user, with respect to the initial version of the tool depicted in Figure 1. Next section depicts the enhancements performed in the GUI to guide the user on the choice of values for the newly implemented simulation characteristics.

3.4 Enhancements in the reporting of simulation results

The intermediate version of the tool has improved the way in which computed results are reported to the user. These enhancements are necessary to eliminate the user expertise on the simulated formal model that the initial version of the tool required. These enhancements include also a graphical aggregation of results in case that the user choice is to execute a *what-if* analysis, which improves the user experience with the tool.

The initial prototype of the tool processed the results obtained by the simulation engine (e.g., GreatSPN) in order to filter the most accurate values and to show them in a view of the IDE. However, these showed results belonged to the domain of the simulated formal model. For instance, using Petri nets and GreatSPN as simulator, the visible results of the tool included the computed mean number of tokens of each Place in the Petri net. In the intermediate version of the tool, the processing of results obtained from the simulation engine of formal models (e.g., GreatSPN) has been extended to convert them to the domain of the quality metrics in which the user is interested. Figure 9 highlights the corresponding part in the *Simulation tool* design that has been extended to implement this enhancement. Figure 10 depicts the way in which these results are shown in the current version and that they belong to the domain of the metric of interest defined by the user (i.e., expected system response time in this example). This functional enhancement increments the satisfaction level of requirement R3.6 and R3IDE.7.

Besides, the availability of *what-if* analysis in the intermediate version has brought a new challenge on how to let the user get information about multiple simulation results in a simple way. The first step towards the solution of this challenge was to implement a new Eclipse View called *Invocation Registry*. This view keeps a list with all the simulations invoked. When a *what-if* analysis is invoked, it gathers the results of each combination of values simulated and shows them hierarchically. This enhancement brings into groups all the results produced by a concrete *what-if* analysis invocation. The outcome is a list of elements, each of them of the same type as depicted in previous Figure 10.

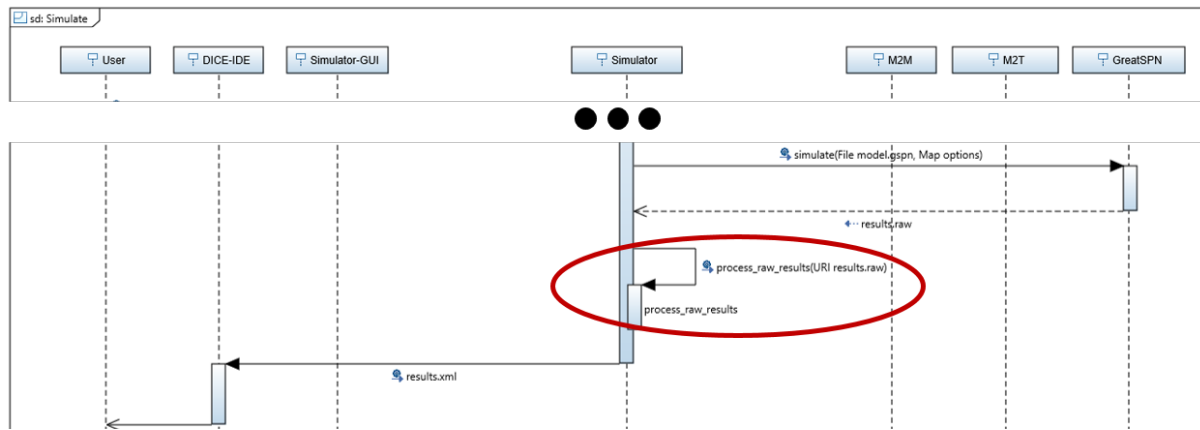


Figure 9: Sequence diagram highlighting the functionality enhancement with respect to the *reporting of simulation results*

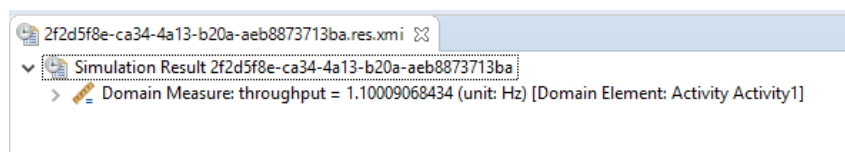


Figure 10: Simulation result of DIA throughput

Although this enhancement brings the possibility to the user to explore all grouped results, s/he still needs to open each single result to see the metric values and mentally devise the tendencies in their values. This process of looking for values in different results hampers the acquisition of a high level view of the tendencies on quality values that a *what-if* analysis can provide. To eliminate such user inconvenience, the intermediate version of the tool incorporates also new methods to produce results in a format that is readable by a plotting package. These new methods automatically get all the measures of interest reported in the single results and build files in the format that a graphical package can read them and draw a graph with their information. Since a great part of this enhancement is related to enhancements in the GUI, more details of the results offered to the user are given in the next section.

4 Updates in the Graphical User Interface

This section provides the enhancements in the GUI of the *Simulation tool*. One of the goals of the GUI enhancements is to provide the user with a graphical manner to manage the configurable options that have been included in the *Simulation tool*. Another goal for modifying the GUI is to improve the user experience during the interaction with the tool. Therefore, the enhanced GUI has been built thanks to the essential feedback that the *Simulation tool* development team has received from project demonstrators. These enhancements are classified into two blocks: enhancements on the GUI to configure a simulation and enhancements on the GUI to visualize the simulation results.

4.1 Enhancements on the GUI to configure a simulation

The GUI that configures a simulation has been extended in order to provide an easy way to the user to set all the new configurable options. The configuration GUI has evolved from the window depicted in Figure 3 to the windows in Figures 11, 12 and 13. These enhancements give graphical support for the satisfaction of requirements R3IDE.1, R3IDE.2, R3IDE.7, R3.1, R3.2 and R3.14.

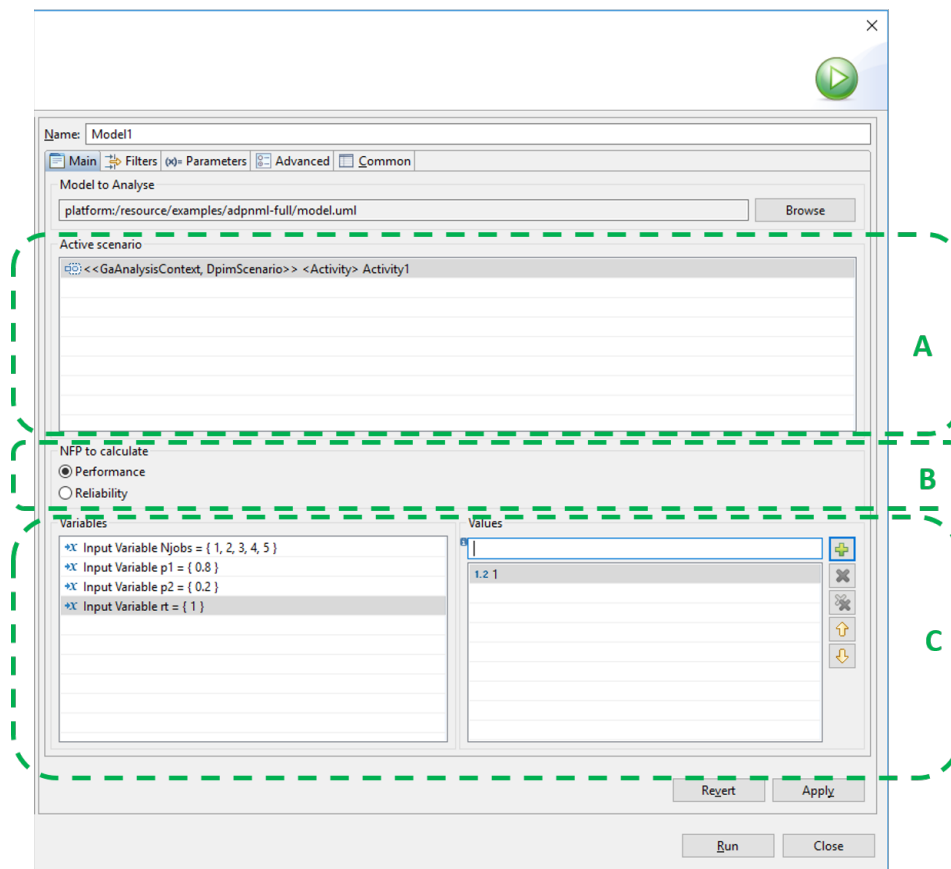


Figure 11: A *Simulation launch configuration* window showing the Main tab.

Figure 11 shows the new GUI to choose:

- The execution scenario to simulate among the multiple that can be now defined in a UML model in the part labeled as A.
- The type of quality evaluation (performance or reliability evaluation) in which the user is interested for the current simulation in the part labeled as B.
- The values of variables defined in the model. To enable *what-if* analysis, it is possible to set a list of values to each of the variables. This is shown in the part labeled as C, where, for instance variable \$Njobs receives five different values, from 1 to 5.

Figure 12 shows the new GUI to choose the quality metric to compute. In the figure, the user has defined both *response time* and *throughput* performance metrics in the UML model, but s/he has unchecked the throughput, indicating that the metric of interest for this simulation is only the response time of the application.

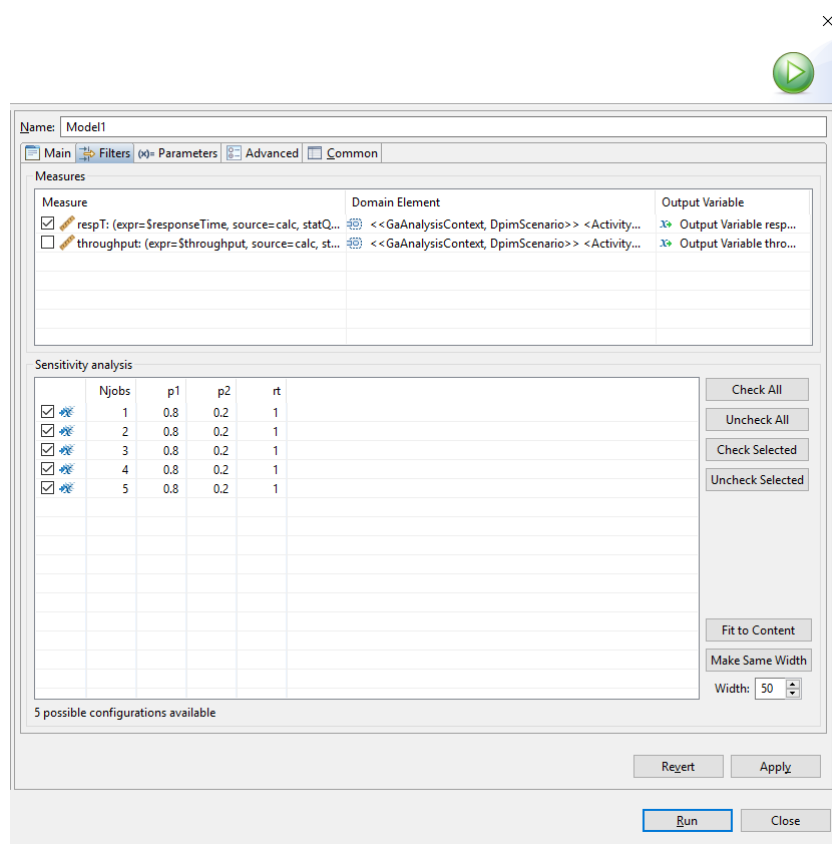


Figure 12: A *Simulation launch configuration* window showing the *Filters* tab.

Figure 13 shows the new GUI to set a Timeout for the execution to avoid indefinitely long simulation times. Concretely, the Timeout value is set in the first field which is called *Maximum simulation execution time*.

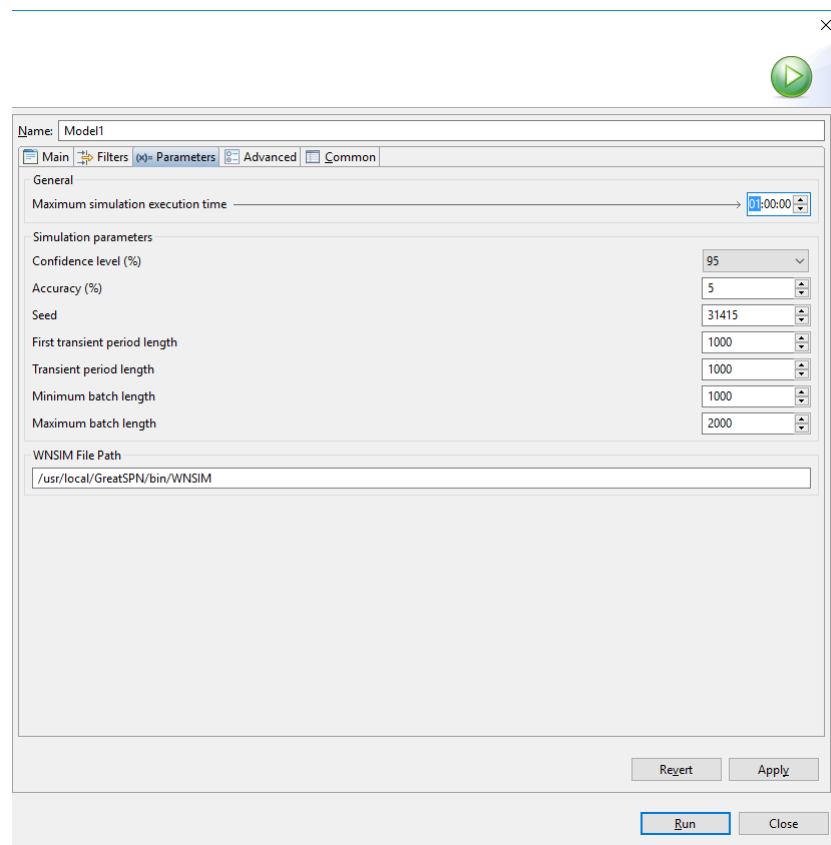


Figure 13: A *Simulation launch configuration* window showing the *Parameters* tab.

4.2 Enhancements on the GUI to visualize the simulation results

The addition of *what-if* analysis capabilities to the *Simulation tool* has provided a rich set of simulation results. This rich set of results has also brought a new visualization challenge: all the simulation results were not easy to see at a single glance of the user, and therefore s/he may lose the big-picture of the results provided by the *what-if* analysis. For this reason, the intermediate version of the tool includes methods to group the *what-if* analysis results with respect to a) one of the input variables over which the analysis has been executed and b) a metric of interest for the user. With these two elements, it is created a 2D plot that reflects the variations on the metric of interest in function of the values of the input variable.

For instance, consider a UML model stereotyped with some DICE profile over which the user desires to execute a *what-if* analysis. Concretely, the user wants to know what would happen with the quality of the system if the arrival rate of requests supported by the DIA varied along a wide set of possibilities from very low to very high values. For doing this, the user has to model the arrival rate of requests as a variable. Besides, the user wants to know the quality of the DIA regarding its expected response time and throughput. Figure 14 depicts the configuration of this analysis. Part (a) gives values for the arrival rate variable, called *rate*, from 0.5 requests per second to 1.3, in increments of 0.1. Part (b) shows the quality metrics that will be computed and the combination of all the variables (in this case, the only variable was the *rate*, so the combination of variables produces just a list with the different values of *rate*).

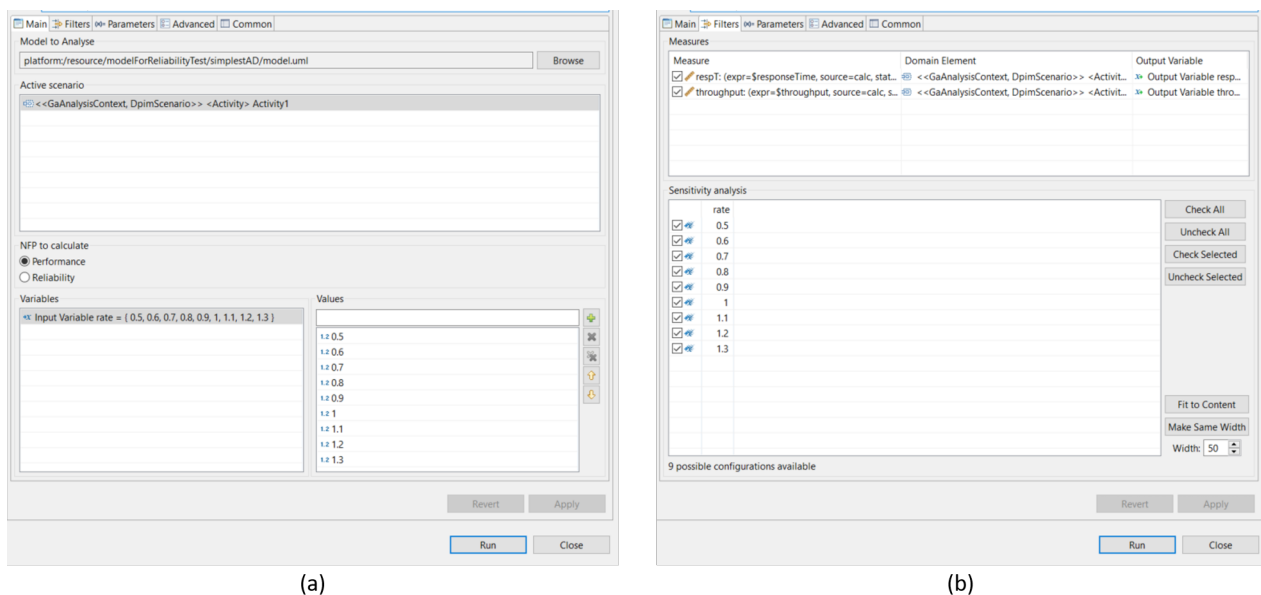


Figure 14: A *Simulation launch configuration* windows defining *what-if* analysis over rate variable and response time and throughput as quality metrics of interests.

Once the model is simulated, the user can see the nine simulation results, one for each value of the rate variable, grouped in the invocation registry. This is depicted in Figure 15.

Simulation	Started	Finished	Status
<<GaAnalysisContext, DpimScenario>> <Activity> Activity1 191519d	Mon Dec 19 12:14:13 ...	Mon Dec 19 12:14:20 ...	Finished
82c4928a-303c-48b2-bce3-38ab94c4daf7	Mon Dec 19 12:14:13 ...	Mon Dec 19 12:14:14 ...	Finished
af1d8ceb-0eab-4ead-b9ba-e00c5242d062	Mon Dec 19 12:14:14 ...	Mon Dec 19 12:14:15 ...	Finished
28fec3f0-8b69-40a4-97f2-dfa44e53ef18	Mon Dec 19 12:14:15 ...	Mon Dec 19 12:14:15 ...	Finished
1bf39fad-32ea-41f4-bd10-34ca478c1e1a	Mon Dec 19 12:14:15 ...	Mon Dec 19 12:14:16 ...	Finished
bb45fa59-37d9-4ffe-a7a6-9998e0167eac	Mon Dec 19 12:14:16 ...	Mon Dec 19 12:14:17 ...	Finished
125f18ee-3744-4951-a001-945dd6a0555e	Mon Dec 19 12:14:17 ...	Mon Dec 19 12:14:17 ...	Finished
5b39f96c-6e5f-4968-ace6-292601e53df7	Mon Dec 19 12:14:18 ...	Mon Dec 19 12:14:18 ...	Finished
cac57539-a1a8-4ac3-b283-2c97214ab58f	Mon Dec 19 12:14:18 ...	Mon Dec 19 12:14:19 ...	Finished
e609aa4f-e9ee-45e3-bccc-dcac30f9f233	Mon Dec 19 12:14:19 ...	Mon Dec 19 12:14:20 ...	Finished

Figure 15: View of the *Invocations Registry* with the results of a *what-if* analysis composed of nine concrete simulations

By double-clicking in each of the concrete results, it opens the results model, analogous to Figure 10, each of them showing the computed response time and throughput. To avoid the user double-clicking one by one each single result and picking the computed values, s/he can now right-click in the container of the nine simulations and choose *Plot results*. This is depicted in Figure 16.

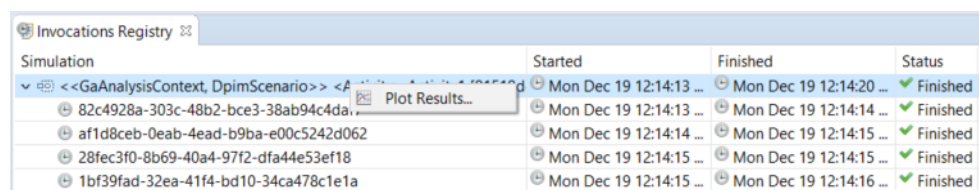


Figure 16: New option *Plot Results* within the *Invocations Registry* view

Then, a wizard opens and the variable and metric of interest can be chosen. These choices will make the x-axis and y-axis of the plot, respectively. Figures 17 and 18 depict these steps. In this case, the user has selected the arrival rate variable for the x-axis (it was the only variable in the model) and the response time as metric of interest for the y-axis. Moreover, it is shown a wizard that asks for a name to the plot and a path in the filesystem to save it, which is depicted in Figure 19.

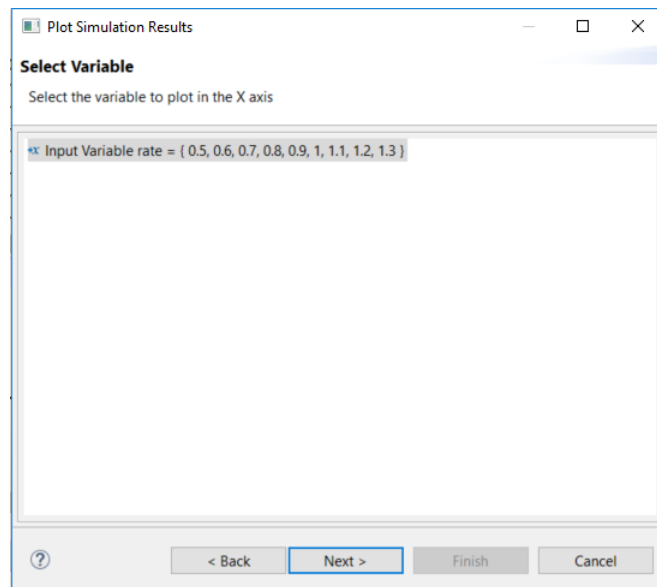


Figure 17: Window to choose the variable to use as x-axis in the plot.

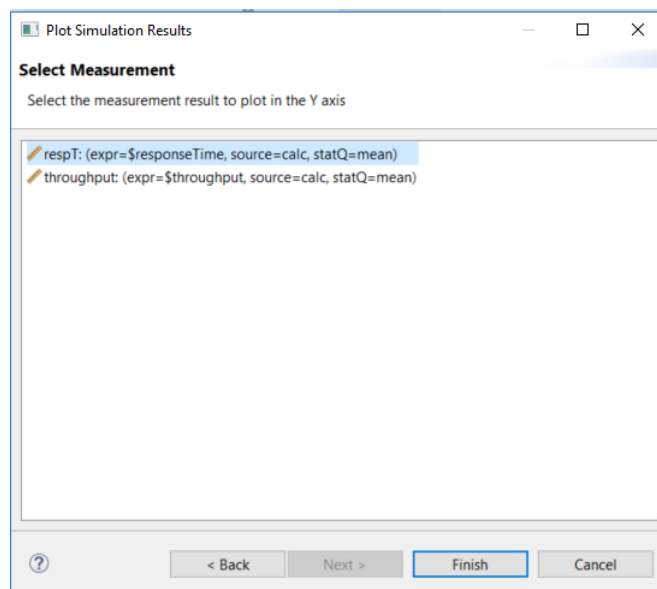


Figure 18: Window to choose the variable to use as y-axis in the plot.

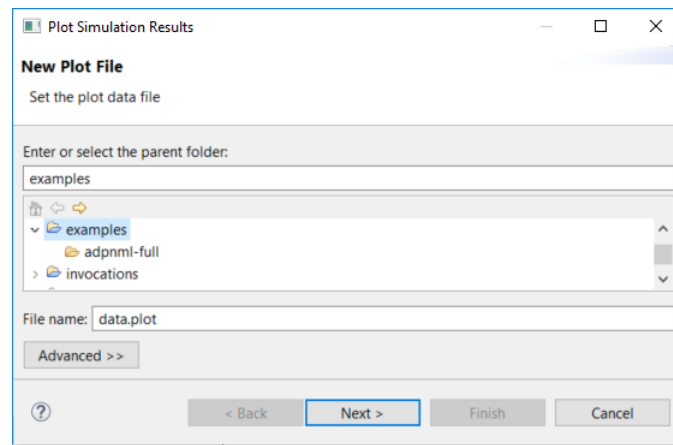


Figure 19: Window to choose the name and path where the plot will be saved.

An example of the result of this plotting is shown in Figure 20. With this view, the user can easily envision the big-picture of the behavior of the DIA response time in function of the arrival rate it receives. This new view increments the satisfaction level of requirement R3IDE.7

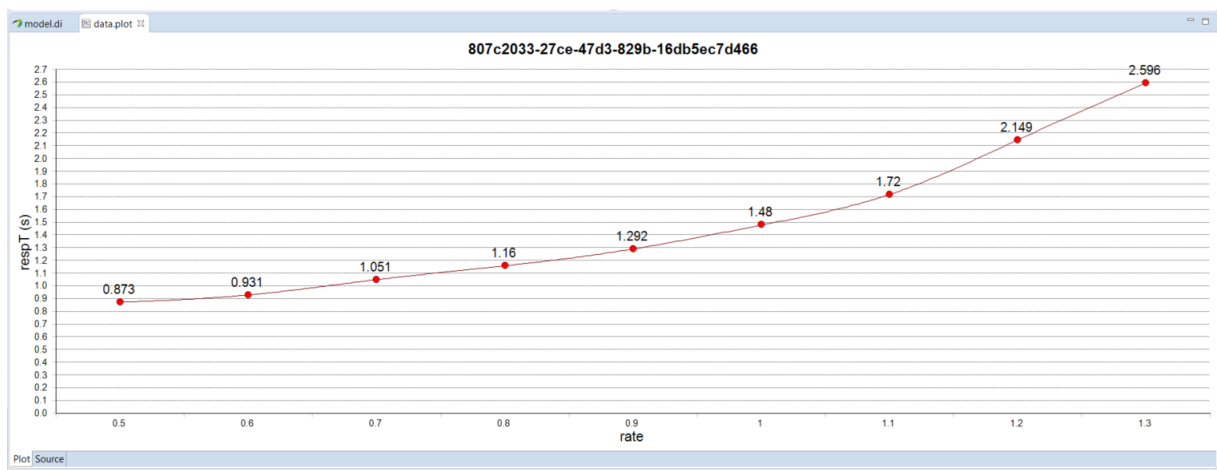


Figure 20: Plot showing *what-if* analysis results with the correlation of how the arrival rate affects the response time.

5 Conclusion

This document has reported the advances achieved in the intermediate version of the *Simulation tool*. This tool is the main outcome of Task T3.2. In its current state, the tool covers all the flow of executions both for the platform independent models (i.e., DPIM level is covered) and for some of the models that are technology specific (i.e., at DTSM level it is possible to simulate Storm models).

Functional enhancements on the simulation characteristics and on the reporting of simulation results presented in Section 3 are tightly coupled with the enhancements on GUI to configure a simulation and enhancements on the GUI to visualize the simulation results presented in Section 4, respectively. Functional enhancement on the quality properties that can be simulated has a partially associated GUI enhancement -i.e., the GUI to configure a simulation-, because the choice among performance or reliability quality property is made during the simulation configuration step. However, there is not a corresponding enhancement in the GUI for the functionality enhancement in the models that can be simulated. The reason is that the UML models to simulate are generated by using Papyrus GUI. Once the DICE UML profiles are imported into the Papyrus execution environment, this Papyrus GUI already offers all the capabilities to generate the models. Therefore, it was only necessary to implement the functionality enhancement in the *Simulation tool* to accept types of DTSM UML models.

Table 3 extends Table 1 with a column with the level of fulfillment of requirements in the current intermediate version of the *Simulation tool*. The meaning of the labels used in the table: (i) ✗ (unsupported: the requirement is not fulfilled by the current prototype); (ii) ✓ (partially-low supported: a few of the aspects of the requirement are fulfilled by the prototype); (iii) ✓ (partially-high supported: most of the aspects of the requirement are fulfilled by the prototype); (iv) ✓ (supported: the requirement is fulfilled by the prototype and a solution for end-users is provided); (v) ▼ (deprecated: the requirement has been deprecated); and (vi) ▲ (new requirement: the requirement has been created in the last update of DICE requirements).

Table 3: Level of compliance with requirements of the intermediate version of the *Simulation tool*

Requirement	Title	Priority	Level of fulfillment initial	Level of fulfillment intermediate
R3.1	M2M Transformation	MUST	✓	✓
R3.2	Taking into account relevant annotations	MUST	✓	✓
R3.4	Simulation solvers	MUST	✓	✓
R3.5	Simulation of hosted big data services	MUST	✗	▼
R3.6	Transparency of underlying tools	MUST	✓	✓
R3.10	SLA specification and compliance	MUST	✓	✓
R3.13	White/black box transparency	MUST	✗	✓
R3IDE.1	Metric selection	MUST	✓	✓
R3IDE.4	Loading the annotated UML model	MUST	✓	✓
R3.12	Modeling abstraction level	MUST	✓	▼
R3.3	Transformation rules	COULD	✓	✓
R3.14	Ranged or extended what if analysis	COULD	✓	✓
R3IDE.2	Timeout specification	SHOULD	✓	✓
R3IDE.3	Usability	COULD	✗	✗
R3IDE.7	Output results of simulation in user-friendly format	COULD		▲✓

The reasons of the declared improvements in the level of fulfillment are the followings:

- R3.1 has passed to be partially-high supported since three new model to model transformations have been included. From DTSM, Storm and Hadoop models to Petri nets for performance evaluation have been implemented and, from DPIM models, a new Petri net transformation has been

implemented for reliability evaluation purposes. This requirement will be fully satisfied when the M2M transformations for performance simulation are implemented for the remaining technologies and when reliability transformation is implemented for DTSM level models.

- R3.2 has passed to be partially-high supported since the transformation tool is able to take: the new annotations that specify characteristics of Storm and Hadoop models (using DICE DTSM::Storm and DTSM::Hadoop profiles, respectively), the annotations that specify dependability characteristics of general systems (using DAM profile), the new annotations regarding the metric in which the user is interested (utilization of resources in DTSM Storm models and reliability in DPIM models). This requirement will be fully satisfied when the remaining technologies (i.e., Spark) have their profile stereotypes and the *Simulation tool* is able to read them and appropriately carry out the simulation of scenarios of these technologies.
- R3.13 has passed to be partially-high supported. This is related to the fulfillment level of R3.1. White/black boxes transparency in modeling has an effect in DTSM level. Since the initial version of the tool could only simulate models at DPIM level, this requirement had not been initialized yet. At DPIM level, performance and reliability annotations in execution scenarios reuse the proposal in MARTE and DAM profiles. Stereotypes provided by MARTE and DAM profiles give complete freedom to the designer to model the system with the desired granularity, from annotating very general steps with a single stereotype to giving a stereotype to each single line of code. Now, being DTSM for Storm and Hadoop models already implemented in the *Simulation tool*, the white/black boxes transparency has been taken into account. The rationale is the following: Hadoop and Storm processes are white boxes, meaning that the designer specifies the steps that are carried out inside such process (e.g., operations executed by Bolt elements in case of Storm, and Map and Reduce operations in case of Hadoop). Figure 6 showed an example of steps in a Storm process. The whole diagram represents a Storm process, so it is a white box. The maximum level of granularity was the specification of StormBolt and StormSpout steps; therefore, these steps are black boxes and what happens inside them remains hidden. These decisions on the level of granularity, on what is modeled as white box and on what remains as black box was taken during the DICE profiles definition task. The *Simulation tool* has implemented its transformations following decisions taken during the profiles definition and using the stereotypes provided by DTSM level profiles. This requirement will be fully satisfied when the rest of technologies that will be supported by the tool will be implemented and follow the same white/black boxes modeling rationale.
- R3IDE.1 has passed to be partially-high supported. Initial version already allowed the definition of several performance metrics in the model, and all of them were evaluated during the simulation. The current intermediate version has increased the number of metrics that the user can define by adding the possibility of defining reliability metrics at DPIM level, and has enhanced the metric selection by allowing the user to choose the metrics in which s/he is interested in each simulation among all the metrics defined in the model. The current state of the *Simulation tool* fully satisfies this requirement.
- R3.14 has passed to be supported. The user can define variables in the model and, during the configuration step, give several values to them. A *what-if* analysis that covers every combination is executed by the tool and its results reported. The current state of the *Simulation tool* fully satisfies this requirement.
- R3IDE.2 has passed to be supported. The user can now define a timeout for the simulation to finish its execution. There is also graphical support for setting this simulation characteristic during the simulation configuration step. Now, if the simulation process has not finished after the maximum simulation execution time, it is aborted and the last intermediate result is kept as final result. The user is also notified in the simulation registry view that the results of the simulation come from an aborted simulation due to timeout expiration.

- R3IDE.7 has been elicited and supported. The user can, through a plot, easily see a set of simulation results obtained from a *what-if* analysis. In this way, the user can devise the tendencies in the behavior of the DIA, infer maximum reachable quality values for the DIA when varying the properties of its elements, or infer the maximum external conditions (e.g., received workload) that the DIA is able to support while satisfying its quality requirements.

As it can be seen, most of the initial (both mandatory and optional) requirements are fully or partially met. Furthermore, we have contributed extensions to the JMT tool [1] to offer a richer simulation capability for DICE, in addition to the support for analysis based on the GreatSPN tool. As JMT is primarily used as a backend for the DICE Optimization Tools, these advances are documented in a different deliverable: “D3.8 DICE Optimization Tools - Initial version” [5] released at M18. A demonstration paper presenting these advances has been recently accepted for demo presentation at ICPE 2017 [2]. We are currently extending JMT to read PNML files, in order to increase compatibility with the Petri net models.

We have decreased the level of fulfillment of R3IDE.4 from fully supported to partially-high supported. The reason of this downgrade is related to the fact that now the *Simulation tool* can work with UML models that define several scenarios (see label A in Figure 11), while the initial version of the tool could deal with UML models that defined only one execution scenario. Therefore, we now consider that the complete satisfaction of R3IDE.4 “Loading the annotated UML model” will be achieved when it will be possible to also load execution scenarios of the currently lacking technologies (i.e., Spark), which will still require some implementation work.

5.1 Further Work

Task T3.2 will still produce an additional deliverable D3.4, the *DICE simulation tools - Final version* at M30. At present, it is not expected any limitation nor delay in the fulfillment of requirements of type MUST. For the requirements whose level of fulfillment have experimented an increment in the intermediate version of the tool -i.e., R3.1, R3.2, R3.13, R3IDE.1, R3.14, R3IDE.2 and R3IDE.7- the previous paragraphs have detailed the work that remains for their fully satisfaction. For the requirements of the *Simulation tool* whose level of fulfillment has not varied in this intermediate version, the work that remains to be carried out until their fully satisfaction is the following:

- R3.4 remains in partially-high supported. The *Simulation tool* is ready to automatically manage more than one simulation solver if they are implemented as Eclipse plugins. At present, only one solver -i.e., the solver that uses GreatSPN simulation engine- is implemented to be used by the tool, so the automatic management of simulation solvers uses always GreatSPN. We mark this requirement as partially-high supported instead of fully supported because, in case that other solvers are required for certain quality properties, their integration could still raise some bugs and therefore the necessity of some more implementation work on the automatic management of solvers.
- R3.10 remains partially-low supported. The required SLA for quality properties can be specified in the UML models of the DIA using DICE profiles, but the compliance check is not implemented yet. This check should compare the results of a simulation with respect to the values defined in the SLA.
- R3.3 remains partially-high supported. The transformations used by the *Simulation tool* are specified in the OMG standard QVT [9] language in separated and dedicated files in a concrete folder. For fully allowing external files that define transformation rules, a new functionality that allows to set at runtime the path of the QVT file that defines the transformations is required. This new functionality should come with an extension in the GUI that allows to set the path of such QVT file in a user-friendly manner.
- R3IDE.3 needs to be fully implemented.

References

- [1] Marco Bertoli, Giuliano Casale, and Giuseppe Serazzi. Jmt: performance engineering tools for system modeling. *SIGMETRICS Perform. Eval. Rev.*, 36(4):10–15, 2009. URL: jmt.sf.net.
- [2] Giuliano Casale, Mattia Cazzoli, Shuai Jiang, Vitor S. Lopes, Giuseppe Serazzi, and Lulai Zhu. Generalized Synchronizations and Capacity Constraints for Java Modelling Tools. Accepted demonstration paper at ACM/SPEC ICPE 2017.
- [3] The DICE Consortium. Requirement specification. Technical report, European Union’s Horizon 2020 research and innovation programme, 2015. URL: http://wp.doc.ic.ac.uk/dice-h2020/wp-content/uploads/sites/75/2015/08/D1.2_Requirement-specification.pdf.
- [4] The DICE Consortium. Requirement specification - companion document. Technical report, European Union’s Horizon 2020 research and innovation programme, 2015. URL: http://wp.doc.ic.ac.uk/dice-h2020/wp-content/uploads/sites/75/2015/08/D1.2_Requirement-specification_Companion.pdf.
- [5] The DICE Consortium. D3.8 dice optimization tools - initial version. Technical report, European Union’s Horizon 2020 research and innovation programme, 2016. URL: http://wp.doc.ic.ac.uk/dice-h2020/wp-content/uploads/sites/75/2016/08/D3.8_DICE-optimization-tools-Initial-version.pdf.
- [6] The DICE Consortium. Dice simulation tools - initial version. Technical report, European Union’s Horizon 2020 research and innovation programme, 2016. URL: http://wp.doc.ic.ac.uk/dice-h2020/wp-content/uploads/sites/75/2016/02/D3.2_DICE-simulation-tools-Initial-version.pdf.
- [7] The DICE Consortium. Requirement Specification M16 update - Deliverable 1.2 Companion Document. Technical report, European Union’s Horizon 2020 research and innovation programme, 2016. URL: <http://wp.doc.ic.ac.uk/dice-h2020/wp-content/uploads/sites/75/2016/05/Requirement-Specification-M16.pdf>.
- [8] The DICE Consortium. Architecture definition and integration plan - final version. Technical report, European Union’s Horizon 2020 research and innovation programme, 2017. *To Appear*.
- [9] OMG. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, Version 1.1, January 2011. URL: <http://www.omg.org/spec/QVT/1.1/>.
- [10] The DICE Consortium. DICE Simulation Repository, Dec., 2015. URL: <https://github.com/dice-project/DICE-Simulation>.
- [11] The Eclipse Foundation & Obeo. Acceleo, Dec., 2015. URL: <https://eclipse.org/acceleo/>.

Appendix A. The DICE-Simulation Repository

This appendix describes the *DICE-Simulation repository* [10]. This repository contains the following projects/plugin-ins:

- es.unizar.disco.core** — This project contains the *Core plug-in*. The *Core plug-in* provides some utility classes for I/O, together with the shared logging capabilities.
- es.unizar.disco.core.ui** — This project contains the *Core UI plug-in*. The *Core UI plug-in* provides UI components that are shared across the different plug-ins contained in this repository, such as file selection dialogs.
- es.unizar.disco.pnconfig** — This project contains the implementation of the *Configuration Model* as an EMF plug-in.
- es.unizar.disco.pnml.m2m** — This project implements the M2M transformations from UML to PNML using QVTo.
- es.unizar.disco.pnextensions** — This project provides some utilities to handle some extensions in PNML models. The PNML standard does not provide support for time, probabilities and priorities in transitions. These features are present in the Generalised and Stochastic Petri nets type or the colored Petri nets of type called Stochastic Well-formed nets. Thus, this plug-in provides the utility methods to handle this information by using the *ToolSpecifics* tags provided by the PNML standard.
- es.unizar.disco.pnml.m2t** — This project contains the Acceleo [11] transformation to convert a DICE-annotated PNML file to a set GreatSPN files.
- es.unizar.disco.simulation.greatspn.ssh** — This project contains the OSGi component that controls a remote GreatSPN instance by using SSH commands.
- es.unizar.disco.simulation** — This project contains the core component that executes a simulation by orchestrating the interactions among all the previous components.
- es.unizar.disco.simulation.ui** — This project contains the UI contributions that allow the users to invoke a simulation within the Eclipse GUI and to graphically visualize the simulation results.
- es.unizar.disco.ssh** — This project provides a simple extension point contribution to access a remote host by issuing the connection data using a local file.
- com.hierynomus.sshj** — This project contains the *sshj - SSHv2 library for Java* as an OSGi-friendly bundle. This module is required by `es.unizar.disco.simulation.greatspn.ssh` to access a remote *GreatSPN* instance using SSH/SFTP.