Performance Testing Modeling: an empirical evaluation of DSL and UML-based approaches

Maicon Bernardino
PUCRS
Av. Ipiranga, 6681, Partenon
Porto Alegre, RS, Brazil
bernardino@acm.org

Elder M. Rodrigues
PUCRS
Av. Ipiranga, 6681, Partenon
Porto Alegre, RS, Brazil
eldermr@gmail.com

Avelino F. Zorzo
PUCRS
Av. Ipiranga, 6681, Partenon
Porto Alegre, RS, Brazil
avelino.zorzo@pucrs.br

ABSTRACT

Performance testing modeling is a relative new research field. Researches investigating how to apply models to document performance testing information essentially started to be reported in the last decade. Motivated by the lack of a standard to represent performance testing information, our research group, in collaboration with an IT company, proposed a UML approach and lately a Domain-Specific Language (DSL) to support performance testing modeling. The goal of this paper is to show how we support our partner company on the decision process to replace UML by a DSL, hence we designed and conducted an experimental study to provide evidence about the benefits and drawbacks when using UML or DSL for modeling performance testing. In this paper, we report an *in vitro* experiment, where the subjects designed *UML models* and *DSL models*, for the purpose of *evaluation* with respect to the *effort and suitability*, from the perspective of the *performance testers and the performance engineers* in the context of *industry and academia* for *modeling performance testing*. Our results indicate that, for performance modeling, effort using a DSL was lower than using UML. Our statistical analysis showed that the results were valid, *i.e.*, that to design performance testing models using our DSL is better than using UML.

CCS Concepts

•Software and its engineering → Software performance; Domain specific languages; •General and reference → *Experimentation;*

Keywords

Experiment; performance testing; domain-specific language

1. INTRODUCTION

Nowadays, software quality has become a very important asset, even for medium/small software development companies. In most of these companies, software quality rely on software testing [8], *e.g.* unit testing, functional testing and performance testing. However, software testing is a time consuming and highly specialized phase in a software development process. For instance, for a company to apply performance testing [6], it requires a team of performance engineers and testers with deep knowledge about the application to be tested, its usage profile, and the infrastructure where it will operate. Moreover, it will also require a performance engineer with great ability on identifying and tracing applications requirements to performance requirements. Another issue faced by performance testing teams is that most companies still write applications requirements in an ambiguous textual format, which could lead to misinterpretation and inconsistencies on the definition of the performance requirements. Moreover, the lack of an unambiguous standard to write the performance test case is another issue to be tackled in performance testing.

In order to mitigate these issues and to improve the communication between development and testing teams, our research group proposed a model-based approach, based on a UML profile, to support the performance testing modeling [11]. After an empirical investigation [13], a partner [1] company started the adoption of our approach, allowing performance teams to get involved in designing test models from early stages of the software development process and supporting the automatic generation of the performance artifacts from the applications models. Despite all the benefits, our daily experience on using UML models to design performance testing revealed that it presents some limitations [2]: (a) most of available UML design tools do not provide support to work with only those UML elements that are needed for a specialized language. Thus, the presence of unused and not required elements may result in an error-prone and complex activity; (b) UML diagrams are restricted to the semantics that is defined by Object Management Group (OMG) [9]. Therefore, in some cases, the available UML elements and their semantics can restrict or even prevent the modeling of some performance features for the Web domain.

To overcome these issues, we have proposed and implemented a Domain-Specific Language (DSL) [5] to the performance domain: Canopus [2]. Canopus provides a graphical and textual DSL to support the design of performance models. Our DSL, unlike the UML Testing profile, is designed to support the performance testing modeling activity and also to be used within MBT tools to generate performance test scripts and scenarios for third-party tools/load

SAC 2016, April 04 - 08, 2016, Pisa, Italy

ACM 978-1-4503-3739-7/16/04…$15.00

DOI: http://dx.doi.org/10.1145/2851613.2851832

[1]Study developed in the context of PDTI 001/2015, financed by Dell Computers with resources of Law 8.248/91.

generators. It is important to notice that our DSL was designed and developed based on our previous experience and on our partner's expertise on performance testing modeling and on applying model-based testing to generate performance scripts and scenarios. Despite all our expertise on performance testing modeling, there was a lack of knowledge about the benefits and drawbacks of using DSL or UML to design performance testing model. Motivated by this lack of knowledge, we planned and conducted an experimental study to provide evidence about the benefits and drawbacks when using UML or DSL for modeling performance testing. These evidences will support our partner company on the decision process on the replacement of UML by Canopus. We understand that some of the limitations, mentioned above, are due to UML being a general propose language, but to the best of our knowledge, there is no work that show that UML performs better, in performance testing, than a DSL.

This paper is organized as follows. Section 2 presents some context relative to the company in place, as well as motivations for the study. Section 3 introduces the experiment instruments. Section 4 presents the experiment design and introduces our research questions. Section 5 describes the execution of the experiment, while Section 6 presents our analysis and interpretation of results. Finally, we present the conclusion and future work.

2. EXPERIMENT CONTEXT

In the past years our research group in performance testing has been investigating, in cooperation with a Technology Development Lab (TDL) of a global IT company, novel approaches and strategies to support performance testing. One of our main research focus is the automation of performance testing process, mainly through the application of Model-based Testing (MBT) [15] approaches and its related languages and modeling notations. In this context, we have proposed and empirically evaluated several approaches [12] [13], notations [4] [14], languages [2] and tools [11].

Basically, all of our research topics are aligned with the industrial needs of our partner company. For instance, our research on performance testing automation wants to cover all the phases of performance engineering to apply model-based performance testing, from models and notations to supporting tools, providing a complete MBT solution in accordance with the company needs. To reach this goal, we initially proposed the use of annotations on UML diagrams to model performance testing. After that, we developed a tool [11] that accepts these models as input and then generates performance scripts and scenarios for third-party tools/load generators. As time progressed and the use/interest in our solutions increased, we received feedback from the testing teams reporting some issues related to the use of UML diagrams to model performance testing. Based on that, we decided to implement a DSL, henceforth Canopus, for the performance testing domain in order to replace the UML models as our performance testing standard modeling notation. The requirements and our design decision on the implementation of Canopus are described elsewhere [2].

The decisions on the replacement of a technology, specially in an enterprise context, must be based on strong evidence. Therefore, to provide evidence about the benefits and drawback on the replacement of UML by Canopus, we designed and setup an empirical experiment in collaboration with the TDL of our partner company.

3. EXPERIMENT INSTRUMENTS

In this section we briefly present the UML approach for modeling performance testing, the Canopus, and LimeSurvey and Moodle [2] used as Systems Under Test (SUT) during the experiment training and execution: (a) **UML profile for performance testing** [11]: allows the use of UML Use Case and Activity diagrams annotated with stereotypes and tagged values to model performance testing. Our approach to model performance information into UML models relies on stereotypes[3] to annotate test scenario information into Use Case diagrams and the test case information into Activity diagrams; (b) **Canopus**: aims to allow performance testing teams to model the activities performed by the application's users and the environment where the applications is hosted. Canopus is composed of three metamodels: *performance monitoring*, *performance scenario* and *performance scripting*. The *performance monitoring* metamodel supports the design of a model to represent the testing environment infra-structure, such as application servers, database servers, and the load generator servers. This metamodel supports the definition of performance counters that will be monitored during the test execution, for every server. The *performance scenario* metamodel supports the modeling of the users workload profiles and the probabilities of execution of test cases by each of these profiles. For every modeled workload profile, it must be instantiated to a *performance workload* metamodel.[4]. A *performance scripting* metamodel is used to model the virtual users behavior, and it must be bound to a user profile; (c) **LimeSurvey**: is an open source Web application that allows non-technical users to quickly create on-line question-and-answer surveys; (d) **Moodle**: is an open source learning management platform that provides a customized virtual environment for educators and students.

4. EXPERIMENT DESIGN

The goal of this experiment is to get evidence about the effort, intuitiveness and effectiveness of using UML or Canopus to create performance testing models. Hence, we stated the following research questions: **RQ1.** *What is the effort to design a performance testing model when using UML or Canopus?* **Null hypothesis**, H_0: effort is the same when using UML and Canopus to design a performance testing model. **Alternative hypothesis**, H_1: the effort is lower when using UML to design a performance testing model than when using Canopus. **Alternative hypothesis**, H_2: the effort is lower when using Canopus to design a performance testing model than when using UML. **RQ2.** *How effective is to design performance testing model when using UML or Canopus?* **RQ3.** *How intuitive/easy is to design performance testing model when using UML or Canopus?*

4.1 Selecting and Grouping Subjects

We chose an *in-vitro* approach, *i.e.* performed in laboratory under controlled conditions (see [16] for terms regarding the experimental research method), to avoid external influences during the execution of the experiment. Therefore, all activities executed by the experiment subjects were per-

[2]https://www.limesurvey.org | https://www.moodle.org/
[3]A detailed description can be found in [4].
[4]A detailed description about the objects that support this metamodel can be found in [2]

formed in a laboratory, under controlled conditions. After the definition of the experiment design, we focused on the selection of the subjects, which is one of the most important activities in an experimental study. Thus, we spent a considerable effort on inviting and selecting subjects from industry and academia. Basically, we focused on the selection of subjects based on the information provided by the partner company about the target subject profiles: junior and senior performance analysts and testers. Therefore, we invited experienced performance engineers and testers from a local company and from a global IT company. We also invited undergraduate students from an university, in order to select subjects with non-industrial experience on performance testing. After selecting the subjects we set the dates to run the experiment sessions - a whole day for both training and execution sessions. Moreover, before the training session we asked the subjects to answer a background survey. From the data extracted from that survey, we randomly assigned the subjects into two groups (randomization). Furthermore, we also kept each group with the same number of subjects and with similar skills (balancing). During the execution session, each group started modeling with a different notation. While *Group 1* started modeling with UML, *Group 2* designed the models using Canopus. In the next phase, *Group 1* started modeling with Canopus and *Group 2* modeled using UML - all subjects executed both treatments (a paired comparison design). After all the subjects had executed the last experiment phase, they answered a post-experiment survey. It is important to highlight that we monitored the time spent by each subject to complete each phase. Thus, the time spent data and the survey results from all the subjects were used to draw the experiment conclusions.

4.2 Instrumentation

The main objects of the instrumentation for our experiment are the performance testing models composed of performance scripts, scenarios and workloads, designed in accordance with both approaches (UML and Canopus) for testing the Moodle application. Furthermore, guidelines were provided for supporting the subjects on the execution session, such as performance requirements, technical specification and use cases. Moreover, we used two tools to support each one of the approaches: Astah Professional [1] version 6.9.0, for modeling the Use Case and Activity diagrams when applying the UML approach, and; MetaEdit+ [7] version 5.1, for designing the graphs supported by the metamodels developed by Canopus [2].

In the training session (using LimeSurvey), the UML approach was introduced to the subjects through an oral presentation using videos to demonstrate how the approach was applied to a real case study. Additionally, we provided the subjects with a manual about UML modeling for performance testing and a detailed instruction on how to use Astah to design performance models. Similarly, Canopus was introduced to the subjects through an oral presentation, which we demonstrated how Canopus could be applied for modeling performance testing. We provided the subjects with a manual about Canopus and also a detailed instruction on how to use MetaEdit+ to design Canopus models. After that, the subjects had to design a user interaction with a Web-based survey. Besides, the subjects could ask open questions for each approach or clarify how to use the tools to design the performance models.

In the execution session, the subjects interacted with another Web application - Moodle. To execute the tasks that compose the experiment, the subjects were introduced to guidelines, describing the test specification. Figure 1 shows one of the use cases used in the experiment. Based on those documents, the subjects had to design the performance models, in accordance with the approach guidelines. Figure 2 presents the UML activity diagram designed in accordance with the Moodle use case described in Figure 1. Figure 3 presents a Canopus Performance Scripting model designed in accordance with the same specification. Due to the heterogeneity of subjects sources, we decided to execute the experiment *in loco*, but *in vitro*, controlling the environment against external interferences. We collected effort metrics for each subject to answer our **RQ1**. To answer **RQ2** and **RQ3** we collected data from the post-experiment survey.

```
Use Case: Add Activity
#Actors: Teacher.
#Finality: Allow teacher to assign activity to course.
#Pre-Condition: The teacher has logged in the Moodle.
1. Select Course
   action: go to "http://www.cepes.pucrs.br/moodle/"
           where id equal 22
2. Enable Editing
   action: submit "Turn editing on" button
3. Click Add a Activity or Resource
   action: click on "Add a Activity or Resource" link
4. Select Assignment Option
   action: select on "Assignment" option and
           submit "Add" button
5. Click
   action: type "Name" text field and
           type "Description" text area
5.A. Save and Display
     action: submit "Save and Display" button
5.B. Save and Return to Course
     action: submit "Save and Return to Course" button
#Pos-Condition: Activity assignment in the course.
                Students able to upload their answers.
```

Figure 1: A Moodle use case specification

4.3 Threats to Validity

In this section, we describe the threats to the experiment validity, and how we work to mitigate them. We adopted the threat classification scheme published by [3], which is divided in four types of threats: (a) **Conclusion validity**: in this experiment context, the small number of subjects, in special the small group of subjects from industry, is a significant threat to the conclusion validity. Others threats to our experiment conclusion validity are the following: *Measures reliability*: this type of threat is related to the researcher bias when analysing the experiment results. To mitigate this threat, the experiment results were independently validated by two researchers involved in the experiment. Moreover, the analysis of quantitative data do not involve human judgment; *Random irrelevancies in the experimental setting*: this type of threat entangles our ability to see a relationship between the treatments and the experiment results. To mitigate this threat, all the experiment activities involving the subjects were executed in a laboratory, isolated from external influences (noise, interruptions, etc.). Moreover, the subjects were not allowed to use mobile

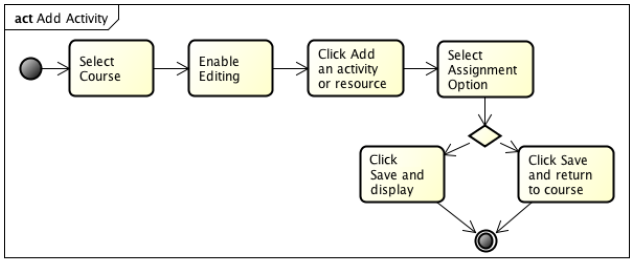

Figure 2: UML activity diagram of the use case specification from Figure 1

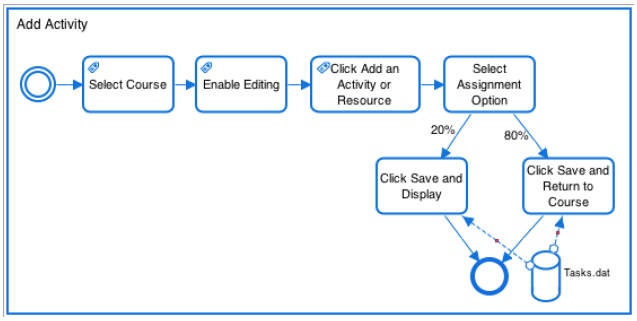

Figure 3: Canopus performance scripting of the use case specification from Figure 1

phones or any other type of communication with the external world; *Random heterogeneity of subjects*: we selected a diverse group of subjects: seven from a large IT company, six from a local company and thirteen undergraduate students. The selection of some subjects with no industrial experience on performance testing, and others with years of experience in software testing and modeling may be a threat to the validation of the experiment results. To mitigate this threat we defined experience on performance testing as blocking variables: inexperienced (IN) and experienced (EX); (b) **Internal validity**: we identified the following threat to the internal validity of our experiment: *Selection*: a survey was applied to assess the knowledge and the experience of subjects and then used to select and group (block) the subjects; (c) **External validity:** we identified the following threat to the external validity of our experiment: *Subjects*: the selection of the subjects that may not be representative to the performance testing community is a threat to the external validity of our experiment. To mitigate this threat, we invited software testing professionals with different levels of expertise in performance testing from two companies. Our decision on inviting undergraduate students was made in order to provide data on the use of the languages for subjects with no knowledge/expertise in performance testing; (d) **Construct validity**: a threat to the construct validity is that we use a single application as SUT, and the subjects modeled a limited set of test cases of the application (mono-operation bias).

5. OPERATION OF THE EXPERIMENT

This section discusses the preparation and execution steps performed during the experiment operation.

5.1 Preparation

Our preparation to the experiment execution includes: to identify what application requirements will be modeled during the training and execution sessions; to prepare the approach guidelines, and; to identify and create practical examples to show during the training session[5]. Furthermore, all laboratory computers were prepared with all the necessary documents, as well as the proper tools (Astah and MetaEdit+). Moreover, we also applied a pre-experiment survey to obtain background information on the subjects. Based on this information, we blocked the subjects and categorized them into two equivalent groups.

[5]Instruments are in http://tiny.cc/SVT16-Canopus

5.2 Execution

The experiment took place in June of 2015. During the training session, both approaches were applied, and for each approach two tasks are executed: performance scenario and scripting modeling. The performance scenario specification of LimeSurvey is composed of two user profiles: a student and a professional respondent. The workload is composed of one thousand virtual users for a testing duration of four hours. This workload executes the "Answer The Survey" script, which is composed of twelve activities that represent each one of the questions from a survey.

During the training session, the subjects were distributed between the treatments and their respective block variables. Hence, twenty-six subjects were assigned to two groups (13 subjects per group). Each group started the execution of one of the two treatments (UML or Canopus). The execution session followed the same systematic approach applied in the training session. Thus, the subjects had to perform two tasks for each treatment, in order to design performance models in accordance to the experiment guidelines for performance testing. One task was to design performance scenario models and another was to design performance scripts models: (a) *Scenario Task* - consists of designing a performance scenario model from scratch, based on the performance requirements. In the context of UML, this model is represented by a Use Case diagram, in which it is composed of two actors: students and teachers. These actors are associated with three use cases, which will be detailed into Activity diagrams. Inasmuch as using Canopus, the subjects had to design the Canopus Performance Scenario. It is important to highlight that a Canopus Performance Scenario has elements that may be decomposed into other elements, *e.g.* Canopus Performance Workload, which is modeled to represent features such as test duration, numbers of virtual users, ramp up and ramp down; (b) *Scripting Task* - the experiment subjects had to design the performance scripts using the modeling strategy described in Section 3, aggregating performance test information to the model. Therefore, using UML, they had to design three Activity diagrams to represent a test case describing the interaction between a virtual user and the SUT. Similarly, using Canopus the subjects had to model the Canopus Performance Scripting to represent the same interaction. Figure 3 shows the Canopus Performance Scripting designed in accordance with the Moodle use case specification described in Figure 1. Additionally, the entire task comprises two more diagrams that represent the *Sign In* and *View Activity* use cases.

6. RESULTS

In this section, we discuss the data collected during the execution phase.

RQ1. *What is the effort to design a performance testing model when using UML or Canopus?* Table 1 presents the summarized effort data (time spent by subjects) to perform each task using each approach, as mentioned in Section 5.2. In the table, the columns Scenario and Scripting present the average time per blocks and treatments, respectively. Based on the results summarized in the table, the average effort using Canopus was lower than with UML in all scenarios, either to experienced or inexperienced subjects. The average time spent to design the performance testing modeling using Canopus was lower than with UML (51.08 min vs 63.69 min).

Table 1: Summarized data of the effort

Effort (minutes)							
Tr.	**Bl.**	**Blocks Avg. Time**			**Tr. Avg. Time**		
		Scen.	**Scr.**	**Total**	**Scen.**	**Scr.**	**Total**
UML	IN	15.69	52.62	68.31	13.62	50.08	63.69
	EX	11.54	47.54	59.08			
DSL	IN	10.54	39.31	49.85	10.23	40.85	51.08
	EX	9.92	42.38	52.31			

Legend - **Tr.**:*Treatments*; **Avg.**:*Average*; **IN**:*Inexperienced*; **EX**:*Experienced*; **Scen.**:*Scenario*; **Scr.**:*Scripting*; **Bl.**:*Blocks*;

Figure 4 depicts the Box-Plot graph of the Scenario Task data set, represented by UML_{Scen} and DSL_{Scen} boxes. In the Scenario Task, the median of execution time with UML was 12.5 minutes and with Canopus it was 10 minutes. Moreover, the UML standard deviation (Std Dev) was 5.02 minutes, against 3.43 minutes for Canopus. It is important to highlight that there is one outlier in the data set for Canopus that took 17 minutes. From another point of view, Figure 4 also presents the Box-Plot graph of the Scripting Task. This task is represented by UML_{Scr} and DSL_{Scr} boxes, where the median time to execute the UML treatment was 50.5 minutes, while for Canopus it was 39.5 minutes. Moreover, the UML Std Dev was 11.84 minutes, greater than the 7.12 minutes for Canopus. Again, notice that there is one outlier in the data set for Canopus that took 23 minutes. Figure 4 also shows the Box-Plot graph of the summarized data set, *i.e.*, the sum of Scenario and Scripting tasks, identified by UML_{Total} and DSL_{Total} boxes. Here, the median of execution time with UML was 62.5 minutes, inasmuch as Canopus was 48.5. It is important to highlight that the Canopus Std Dev (9.46 minutes) represents 66.8% of variation of the UML treatment (14.16 minutes). Once again, notice that there is one outlier in the data set for Canopus treatment that took 28 minutes. Figure 5 presents the box-plot graph grouped by blocks. The median of execution time with UML_{IN}, UML_{EX}, DSL_{IN} and DSL_{EX} were, respectively, 65, 62, 48 and 49 minutes. Moreover, the Std Dev for each block was, respectively, 14.65, 12.53, 10.11 and 9.0.

As for hypothesis testing, we used the PortalAction statistical package [10] integrated with MS Excel to test our hypothesis from the collected data sets. We performed the Kolmogorov-Smirnov [10] test to verify the normality of data distribution. In this context, we followed the best practice in Statistics and chose a significance level of $\alpha = 0.05$. Although, almost all results had a normal distribution, there was one exception, *i.e.*, the Scenario data set, that showed a p-value (0.355835174) greater than α. For this reason, we assumed that the distribution was not normal. Therefore, we applied a non-parametric test: Wilcoxon [10] signed rank test. We applied a non-parametric test since it uses the median instead of average as used in parametric test, and this solves the problem with outliers. For each data set (Scenario, Scripting and Total), we applied the statistical test to the paired samples, *i.e.* to compare the effort spent to model performance using UML or Canopus (**RQ1**). As presented in Table 2, for all samples pairs, the results of the Wilcoxon test reject the H_0 hypothesis. Therefore, we can conclude that there is, statistically, a noticeable difference in effort to design a performance testing model when using UML and Canopus. Thus, for all data sets we confirmed the alternative hypothesis H_2.

Table 2: Wilcoxon signed rank test results

Treatment	Scenario	Scripting	Total
UML/Canopus	0.000603644	0.007055169	0.000124493

RQ2. *How effective is to design performance testing model when using UML or Canopus?* **RQ3.** *How intuitive/easy is to design performance testing model when using UML or Canopus?* After designing the performance models using both approaches, the subjects answered a survey composed of: (a) statements to survey how much they concur with our Canopus features; (b) open questions to extract their opinions. The answers can be seen in the frequency diagram shown in Figure 6. As can be seen in the figure, the statement most accepted is that the Canopus DSL has **Expressiveness**[6] (61.5% SA, 27% A and 11.5% NAD), followed by that it has **Representativeness**[7] (50% SA, 34.6% A, 15.4% NAD) and **Easy to Design** (30.8% SA, 65.4% A, 3.8% NAD). The statement that received the worst mark was with respect to **Intuitiveness** (53.9% A, 26.9% NAD and 19.2% D).

The main advantages on the use of the UML approach, by the subjects point of view were: (a) *It was easier because I already had some experience with UML*; (b) *I already had used Astah tool to design UML, so it was easier to use*; (c) *There are tons of documentation and discussion groups on the Internet.* The subjects also pointed some disadvantages on the use of UML: (a) *Lack of interaction/reuse of models with parameters*; (b) *Higher probability of making a syntax error when typing the tag values*; (c) *Some performance testing information cannot be easy to design with UML.* Similarly, the subjects reported advantages on the use of Canopus: (a) *Promotes the code reuse in all modeling phases. Easy and fast understanding of the interface*; (b) *It is intuitive to realize the relations that are being performed. The scenario creation is intuitive and the probability of making a syntax error is smaller*; (c) *Allows to add many performance testing information to the models. Adding parameters is much easier.* They pointed out the following disadvantages: (a) *Learning curve - at the beginning it was complex to use. It was difficult to understand which element should be used*; (b) *Lack of mechanism to duplicate an element.*

7. CONCLUSION

[6]Expressiveness: defines the quality of graphical elements composed by color, size, brightness, shapes, and texture.

[7]Representativeness: whether the graphical elements represent or not the performance testing domain.

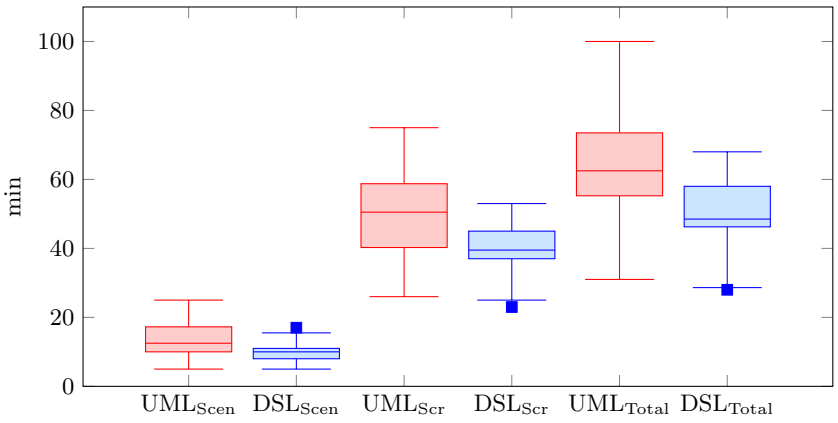

Figure 4: Boxplot - treatments per task

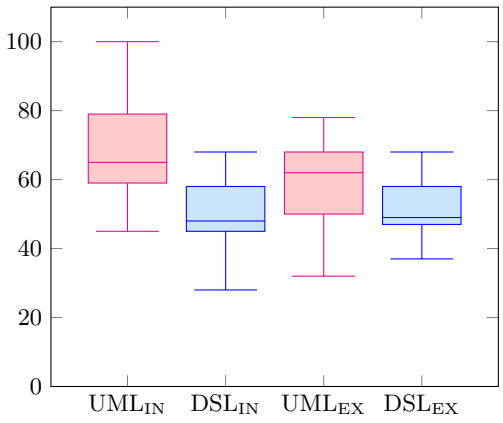

Figure 5: Boxplot - treatments per block

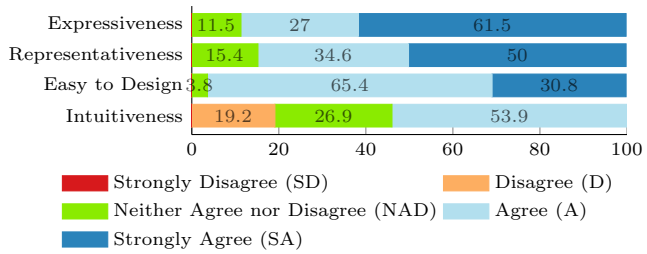

Figure 6: Frequency diagram of the Canopus

In this paper, we have presented an *in vitro* experimental study for evaluating our DSL, and for providing evidence about the benefits and/or drawbacks when using UML or DSL approaches for modeling performance testing. The experimental results indicate that the use of Canopus reduces the effort required to create models when compared to UML. Moreover, the experiment subjects pointed out that Canopus has more expressiveness, representativeness, and is easier to design than UML. Therefore, these findings can provide support to our partner company on the decision process to replace UML by Canopus for modeling performance testing. We are aware that the sample size that was used to base some of our conclusions is not geographically relevant. Moreover, a group of subjects did not have an adequate experience level or knowledge on performance testing. Therefore, we designed our experiment protocol with the intention to replicate it in the future to collect more results. Moreover, based on the achieved findings, testimonials from experiment subjects and our lessons learned to conduct this experiment, we intend to improve our DSL, ensuring the evolution of Canopus for a new version of graphical and textual language, as well as new features.

8. REFERENCES

[1] Astah. Astah Professional. Available in: http://astah.net/, 2015.

[2] M. Bernardino, A. F. Zorzo, E. Rodrigues, F. M. de Oliveira, and R. Saad. A Domain-Specific Language for Modeling Performance Testing: Requirements Analysis and Design Decisions. In 9^{th} *ICSEA*, pages 609–614, 2014.

[3] T. D. Cook and D. T. Campbell. *Quasi-Experimentation: Design and Analysis Issues for Field Settings*. Houghton Mifflin, 1979.

[4] L. T. Costa, R. Czekster, F. M. Oliveira, E. M. Rodrigues, M. B. Silveira, and A. F. Zorzo. Generating performance test scripts and scenarios based on abstract intermediate models. In 24^{th} *SEKE*, pages 112–117, 2012.

[5] M. Fowler. *Domain Specific Languages*. Addison-Wesley Professional, 1st edition, 2010.

[6] J. Meier, C. Farre, P. Bansode, S. Barber, and D. Rea. *Performance Testing Guidance for Web Applications: Patterns & Practices*. Microsoft Press, 2007.

[7] MetaCase. MetaEdit+. Available in: http://www.metacase.com/mep/, 2015.

[8] G. J. Myers and C. Sandler. *The Art of Software Testing*. Wiley, New York, NY, USA, 2004.

[9] OMG. Unified Modeling Language. Available in: http://www.uml.org/, 2015.

[10] Portal Action. System action statistical package. Available in: http://www.portalaction.com.br, 2014.

[11] E. Rodrigues, M. Bernardino, L. Costa, A. Zorzo, and F. Oliveira. PLeTsPerf - A Model-Based Performance Testing Tool. In *IEEE* 8^{th} *ICST*, 2015.

[12] E. M. Rodrigues, F. M. de Oliveira, L. T. Costa, M. Bernardino, A. F. Zorzo, S. d. R. S. Souza, and R. Saad. An empirical comparison of model-based and capture and replay approaches for performance testing. *EMSE*, pages 1–30, 2014.

[13] E. M. Rodrigues, R. S. Saad, F. M. Oliveira, L. T. Costa, M. Bernardino, and A. F. Zorzo. Evaluating Capture and Replay and Model-based Performance Testing Tools: An Empirical Comparison. In 8^{th} *ACM/IEEE ESEM*, pages 9:1–9:8, 2014.

[14] M. B. Silveira, E. M. Rodrigues, A. F. Zorzo, H. Vieira, and F. Oliveira. Model-based automatic generation of performance test scripts. In 23^{rd} *SEKE*, pages 258–263, 2011.

[15] M. Utting and B. Legeard. *Practical Model-Based Testing: A Tools Approach*. Morgan Kaufmann, 2006.

[16] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, and B. Regnell. *Experimentation in Software Engineering*. Springer, 2012.