Empirical Evaluation of UML Modeling Tools–A Controlled Experiment

Safdar Aqeel Safdar[1(✉)], Muhammad Zohaib Iqbal[1,2], and Muhammad Uzair Khan[1]

[1] Software Quality Engineering and Testing Lab (QUEST),
National University of Computer and Emerging Science, Islamabad, Pakistan
safdar.aqeel@questlab.pk, {zohaib.iqbal,uzair.khan}@nu.edu.pk
[2] Interdisciplinary Centre for Security, Reliability and Trust, Luxembourg, Luxembourg

Abstract. Model driven software engineering (MDSE) has shown to provide mark improvement in productivity and quality of software products. UML is a standard modeling language that is widely used in the industry to support MDSE. To provide tool support for MDSE, a large number of UML modeling tools are available, ranging from open-source tools to commercial tools with high price tag. A common decision faced while applying UML in practice is the selection of an appropriate tool for modeling. In this paper we conduct a study to compare three of the well-known modeling tools: IBM Rational Software Architect (RSA), MagicDraw, and Papyrus. In this study we conducted an experiment with undergraduate and graduate students. The goal is to compare the productivity of the software engineers while modeling with the tools. We measure the productivity in terms of modeling effort required to correctly complete a task, learnability, time and number of clicks required, and memory load required for the software engineer to complete a task. Our results show that MagicDraw performed significantly better in terms of learnability, memory load, and completeness of tasks. In terms of time and number of clicks, IBM RSA was significantly better while modeling class diagrams and state machines when compared to Papyrus. However no single tool outperformed others in all the modeling tasks with respect to time and number of clicks.

Keywords: Model driven software engineering · UML · Modeling tools · Controlled experiment · Empirical software engineering

1 Introduction

A large number of commercial and open-source tools are available to support UML modeling, including IBM Rational Software Architect (RSA), MagicDraw, Papyrus, Enterprise Architect, Visual Paradigm, Rational Rose, and Argo UML. The choice of selecting a modeling tool has a great impact on the overall success of MDSE [1, 2, 3, 4]. The available tools not only vary greatly in their price tag, but also vary in terms of their features, and can greatly impact the productivity of an engineer. Feature list and price tag can easily be compared directly, but the other aspects that impact productivity need thorough empirical evaluations. Such evaluations are not easy to conduct for end users due to limited resources and restricted access to full version of the available tools, in particular when evaluating commercial tools.

G. Taentzer and F. Bordeleau (Eds.): ECMFA 2015, LNCS 9153, pp. 33–44, 2015.
DOI: 10.1007/978-3-319-21151-0_3

To evaluate productivity, we compared the tools in terms of modeling effort required to correctly complete a task, learnability, and memory load required for the engineer to complete a task. We consider these factors to be important as they directly or indirectly affect the productivity of software engineer.

Controlled experiments are widely used in the domain of software engineering for the comparison and empirical evaluation of different techniques and tools [5, 6]. We have used controlled experiment for comparison of UML modeling tools. For our controlled experiment we selected one open source and two commercial tools for comparison, which include: IBM Rational Software Architecture, Papyrus, and MagicDraw, whereas Enterprise Architect is used to train the participants. These are three of the widely used modeling tools that are based on eclipse and fully support UML 2.0 and EMF compatible XMI export [1, 3].

In this paper we conducted a controlled experiment with multiple participants to compare the productivity of selected UML modeling tools. The experiment is conducted with 30 students at National University of Computer and Emerging Sciences, Islamabad, Pakistan. In the experiment, we gave three of the most widely used UML diagrams [7], i.e., class diagram, sequence diagram, and state machine to model in an assigned tool and a questionnaire form to fill at the end of the activity. The questionnaire form contains the details related to starting time and completion time corresponding to each diagram, learnability measure, memory load measure, and guidelines for completing the activity. Our results show that MagicDraw performed significantly better in terms of learnability, memory load, and completeness of tasks.

The rest of the paper is organized as follows: Section 2 summarizes the literature review. Section 3 presents the details related to planning of the experiment. Section 4 provides the results and analysis of our experiment corresponding to each research question. In Section 5 we provide overall discussion. Section 6 highlights the threats to validity and finally in Section 7 we conclude our work.

2 Related Work

This section discusses the existing literature related to MDSE modeling tools evaluation in software engineering.

After a thorough search, we found a total of seven studies in literature that compare and evaluate modeling tools. Eichelberger *et al.* [3] conducted a comparative survey in which around 75 modeling tools were evaluated. They derived a hierarchal set of features from UML superstructure document. The purpose of the survey was to assess the compliance with standard UML and classify the tools into different compliance levels. Khaled [2] described a set of desired features of UML modeling tools and compared four modeling tools (i.e., Rational Rose, ArgoUML, MagicDraw, and Enterprise Architect) based on these features. Sengoz *et al.* [8] compared two design modeling tools (UML-SPT-Rhapsody and UML-RT-Rose RT) for the real time software modeling based on their features. Rani and Garg [9] compared four UML modeling tools, i.e., ArgoUML, StarUML, Umbrello UML Modeller, and Rational Rose based on their features. Heena and Ranjna [4] compared five modeling tools, i.e., Rational Rose, MagicDraw, ArgoUML, UMLet and Visual paradigm based on their features.

In [10], the authors compare the UML modeling tools in terms of time required for modeling and subjective opinion of participants regarding tools' features. They also

applied goals, operator method, and selection (GOMS) technique to measure the modeling effort required. This study was conducted in 2005.

Almost all of the above discussed studies either compare the tools merely based on their features [2, 4, 8, 9] or are surveys with an intent to classify the tools [3, 11]. There is only one reported experiment to compare UML modeling tools [10]. We believe that such an empirical study is crucial to help the software engineers select a particular tool. In our experience [1, 12], selection of a modeling tool has a great impact on the successful implementation of MDSE in industry. The experiment in [10] was conducted in 2005. Since then the UML as a standard has grown enormously and similarly the corresponding tool support has evolved. The study in [10] cannot be used today to help software engineers in selection of a modeling tool.

In our experiment, we not only measure the time required for modeling (as in [10]), but we also evaluate the tool in terms of how much of the diagrams developed by the participants are complete and correct. Moreover, we also compared the tools in terms of learnability and the memory load required for the engineer to complete a task.

3 Experiment's Planning

In this section, we discuss the details related to the experiment's planning phase. These details are in accordance with the guidelines for reporting experiments provided by Wohlin *et al.* [13]. Section 3.1 provides goals, research questions, and hypotheses, Section 3.2 presents the details related to participants' selection for the experiment whereas in Section 3.3 we provide the details related to experiment material, i.e., case study and questionnaire form. Section 3.4 defines the dependent and independent variables of the experiment. Section 3.5 discusses the experiment design, whereas Section 1.1 provides the details related to training of participants. Finally in Section 3.7 we discuss the details related to statistical tests that we apply to analyze the experiment's data.

3.1 Goals, Research Questions, and Hypotheses

The objective of our experiment is to compare the productivity of three of the widely referred UML modeling tools: IBM RSA, MagicDraw, and Papyrus. We compare the productivity when working with these tools in terms of modeling effort, learnability, and memory load. To evaluate the modeling effort we consider two factors: 1) the time required for completing a given task, 2) the number of clicks required for completing a given task. We also consider completeness of the diagrams developed as part of the modeling effort. Following are the research questions for the experiment conducted in order to evaluate modeling effort, learnability, and memory load:

RQ1: *Which tool among IBM RSA, MagicDraw, and Papyrus is better with respect to modeling effort?*

To answer RQ1 we compare the UML tools in terms of effort required to complete the given modeling tasks correctly. We further divide RQ1 into two sub research questions as follows:

RQ1.1: *In which modeling tool, among IBM RSA, MagicDraw, and Papyrus, the users were able to correctly model more complete diagrams?*

We measure completeness as the percentage of modeling elements modeled correctly by the user for a particular task compared to the total number of modeling elements in the reference model for that task. Measuring completeness will allow us to determine

the modeling tool in which the participants were able to correctly model more modeling elements in the given time. By correctness we mean that modeler used the correct meta-elements for modeling a given diagram. For example to add a constraint on a class, modeler should use the UML *Constraint* meta-element. If the modeler used a *Comment* or a *Note* to model the constraint it will be incorrect.

RQ1.2: *Which tool among IBM RSA, MagicDraw, and Papyrus is better with respect to time and number of clicks required for modeling?*

We are interested in comparing the tools in terms of time and number of clicks required to complete the given modeling tasks.

RQ2: *Which tool among IBM RSA, MagicDraw, and Papyrus is better with respect to learnability?*

Learnability, i.e., how easy is it to learn, is an important characteristic of a modeling tool. RQ2 aims to compare the learnability of the three modeling tools.

RQ3: *Which tool among IBM RSA, MagicDraw, and Papyrus is better with respect to memory load?*

Memory load refers to the amount of information a user needs to keep in mind while completing a specific modeling task.

Before the execution of experiment, we assume that there is no difference in terms of any factor, which leads to two-tailed null hypotheses of our research questions as provided in Table 1.

Table 1. Null hypotheses

Research question	Hypotheses
RQ1.1	H1.1.1: There is no difference between IBM RSA and MagicDraw in terms of completeness.
	H1.1.2: There is no difference between IBM RSA and Papyrus in terms of completeness.
	H1.1.3: There is no difference between MagicDraw and Papyrus in terms of completeness.
RQ1.2	H1.2.1: There is no difference between IBM RSA and MagicDraw in terms of effort required for modeling.
	H1.2.2: There is no difference between IBM RSA and Papyrus in terms of effort required for modeling.
	H1.2.3: There is no difference between MagicDraw and Papyrus in terms of effort required for modeling.
RQ2	H21: There is no difference between IBM RSA and MagicDraw in terms of learnability.
	H22: There is no difference between IBM RSA and Papyrus in terms of learnability.
	H33: There is no difference between MagicDraw and Papyrus in terms of learnability.
RQ3	H31: There is no difference between IBM RSA and MagicDraw in terms of memory load.
	H32: There is no difference between IBM RSA and Papyrus in terms of memory load.
	H33: There is no difference between MagicDraw and Papyrus in terms of memory load.

3.2 Participants

The experiment was conducted in two sessions with two different groups of participants having different skill sets. The first group comprises of undergraduate students who have taken just one modeling course and have a working experience with only one modeling tool, i.e., Enterprise Architect. The second group includes graduate students who have working experience with more than one modeling tools.

In the first session, the experiment was conducted with 18 undergraduate students of final year in computer science at National University of Computer and Emerging Sciences, Islamabad, Pakistan. The university has a well-defined unbiased grading policy and we have used students' grades in the undergraduate course on software modeling to form three different blocks: 1) students with A grade, 2) students with B grade, 3) students with C grade. We have selected the grades of this course for blocking because this is the only course related to UML modeling that they have studies. In this course, students study UML language and complete their semester project using Enterprise Architect for modeling. We organized the students in three balanced groups where each group contains an equal number of students from each block.

In the second session, the experiment was conducted with 12 students, who are currently enrolled in "Advance Software Engineering" course of their MS in software engineering at the same university. All students have experience with different modeling tools. We divided them into three groups on the basis of their experience with the modeling tools.

3.3 Experiment Material

Case Study: The case study used for the experiment contains three diagrams, i.e., class diagram, sequence diagram, and state machine diagram. Since it is not feasible to include all UML diagrams in the case study due to limited resources, therefore we have only selected three of the most commonly used UML diagrams [7]. These diagrams have been used for industrial case studies ranging from automated code generation from models [14], to model based testing [15] and model driven refactoring [16]. We could not find a case study, which contains all UML class diagram elements, so we extended the existing diagrams and combined them in one case study. The sequence diagram, we use is taken from "UML 2 Toolkit" [17] and altered by adding missing sequence diagram elements, e.g., constructor, destructor, and asynchronous call. The state machine is taken from "Testing Object-Oriented Systems, Models, Patterns, and Tools" [18] and altered to add missing state machine diagram elements, e.g., OCL constraints and *do* activity. The complete set of diagrams can be found in [19].

Questionnaire Form: The questionnaire form is carefully designed to measure the learnability and memory load of UML modeling tools. To measure the time required for modeling we asked the participant to note down the starting time and completion time for each diagram. We also run an automated script to capture the number of clicks for each diagram. To measure the completeness, we checked the modeled diagrams against their reference diagrams. The questionnaire form contains personal data of individuals, information about their experience with modeling, information about starting time and completion time of each task (diagram), general guidelines, guidelines for each task, and information about the learnability and memory load of UML modeling tool.

3.4 Independent/Dependent Variables

This section presents the independent and dependent variables involved in our experiment. The independent variables include tools and diagrams whereas dependent variables include completeness, time and number of clicks required for modeling, learnability, and memory load.

Tools: We selected three UML modeling tools, i.e., IBM RSA, MagicDraw, and Papyrus as discussed above.

Diagrams: We selected three most commonly used diagrams class diagram (CD), sequence diagram (SD), and state machine (SM) diagram in the domain of MDSE [7].

Completeness: To measure the overall completeness of modeling tasks we measure the completeness of each diagram. The formula for measuring the completeness of each diagram is given below:

$$Completeness_{CD,SE,SM} = \sum_{i=1}^{n}(Completeness\ of\ each\ type\ of\ Construct_i) \quad (1)$$

Here *n* shows the maximum types of constructs included in a particular diagram. For example in case of class diagram constructs included are classes, enumeration and interface, constraints, cardinality, and relationships. In the same fashion we measured the completeness for sequence diagram and state machine diagram using the above mentioned formula. To compute the completeness of each type of construct we measure the fraction of total instances of a particular type, which are modeled.

Time required for modeling: To measure the time required for modeling we computed the time taken for the participants to model each diagram. The formula for computing the time is given in equation (2).

$$Time = Completion\ Time - Starting\ Time \quad (2)$$

Number of clicks required for modeling: First we captured the number of clicks using an automated script and then used a simple Java program to count the total number of clicks corresponding to each diagram.

Learnability: Learnability refers to the ease with which a novice user can utilize a particular tool to complete a specific task. To measure the learnability of a tool we asked participants to rate it on a scale of five after completing the given modeling tasks.

Memory load: By memory load we mean to keep least information in mind to complete a specific task in the given modeling tool. To measure the memory load of a tool we asked participants to rate it on a scale of five after completing the given tasks.

3.5 Experiment Design

The experiment design summarized in Table 2. We divided the students into three groups and labeled them as group 1, group 2 and group 3. Since these groups are already balanced and have overall same skills set (Section 3.2), therefore, we assigned group 1, group 2 and group 3 to IBM RSA, MagicDraw, and Papyrus respectively. Each group was asked to draw the three given diagrams (class diagram, state machine, and sequence diagram) in the assigned tool.

Table 2. Experiment design

Diagrams to draw	UML modeling tools		
-	IBM RSA	MagicDraw	Papyrus
Class diagram	Group 1	Group 2	Group 3
State machine	Group 1	Group 2	Group 3
Sequence diagram	Group 1	Group 2	Group 3

3.6 Training

All the undergraduate students have studied a course on software modeling in which they were required to complete a semester project that involved creation of a number of UML diagrams using Enterprise Architect. Therefore, they did not need any particular training in using Enterprise Architect. The graduate students have experience of using a variety of modeling tools. Thus to make sure that all the participants have similar expertise of using Enterprise Architect, we gave a 45 minutes training to the graduate students on Enterprise Architect. After the training, we gave them a home assignment to practice class diagram, sequence diagram, and state machine diagram using Enterprise Architect.

3.7 Selection of Statistical Tests

We apply statistical tests in order to assess whether there is a significant difference among the tools being compared (Section 3.1). For this purpose, we used non-parametric Mann Whitney U-test for skewed and parametric T-test for normal data distribution of experimental results [20]. For all statistical tests reported in this paper we have used the recommended significance level of $\alpha = 0.05$. Since our data sample (i.e., the data obtained from results) is small, we used Shapiro-Wilk test to check the distribution of data samples. These tests are commonly applied in literature for analyzing results of such controlled experiments [20].

4 Results and Analysis

This section presents the results and discussion related to each research question defined in Section 3.1. In Table 3 we provide the descriptive statistics for all productivity measures, i.e., completeness, number of clicks and time required for modeling, learnability, and memory load. The raw results of the experiment can be found in [19].

Table 3. Descriptive statistics for all productivity measures

Measures	Diagrams	RSA-MD*		RSA-P*		MD-P*	
		P-value	A12	P-value	A12	P-value	A12
Completeness	CD*	0.191	0.5	*0.020*	*0.62*	0.378	0.5
	SM*	*2.5e-5*	*0.34*	0.278	0.5	*0.006*	*0.62*
	SD*	*0.020*	*0.37*	0.326	0.5	*0.001*	*0.68*
	Overall	*0.025*	*0.43*	0.252	0.5	*0.001*	*0.60*
Number of Clicks	CD	0.093	0.5	*0.008*	*0.16*	1.00	0.5
	SM	0.177	0.5	*0.025*	*0.3*	0.266	0.5
	SD	0.755	0.5	0.730	0.5	0.648	0.5
	Overall	0.962	0.5	0.756	0.5	0.775	0.5
Time	CD	0.968	0.5	*0.022*	*0.27*	0.105	0.5
	SM	0.385	0.5	*0.003*	*0.4*	0.324	0.5
	SD	0.076	0.5	0.114	0.5	0.604	0.5
	Overall	0.231	0.5	0.656	0.5	0.299	0.5
Learnability	-	*0.043*	*0.31*	*0.014*	*0.72*	*3.89e-5*	*0.85*
Memory load	-	0.123	0.5	0.165	0.5	*0.007*	*0.85*

* CD: Class diagram, SM: State machine, SD: Sequence diagrams, MD: MagicDraw, P: Papyrus, RSA: Rational Software Architect, in comparison of A-B where P-value < 0.05, if A12 < 0.5 then B is better than A whereas if A12 > 0.5 then A is better than B.

4.1 Completeness

The data sample for completeness is not normally distributed; therefore we apply the non- parametric Mann-Whitney U-test to check the truthfulness of our hypotheses and Vargha and Delaney's A12 statistic to find the direction and magnitude of difference between the tools. We compared the tools in pairs. The statistical results of the tests for the completeness measure are provided in Table 3. Fig. 1 shows the average percentage of completed tasks in the three tools corresponding to each diagram.

According to the results of statistical tests corresponding to completeness (Table 3), there is no significant difference between RSA-MagicDraw and Magic-Draw-Papyrus in terms of completeness for modeling class diagrams, though

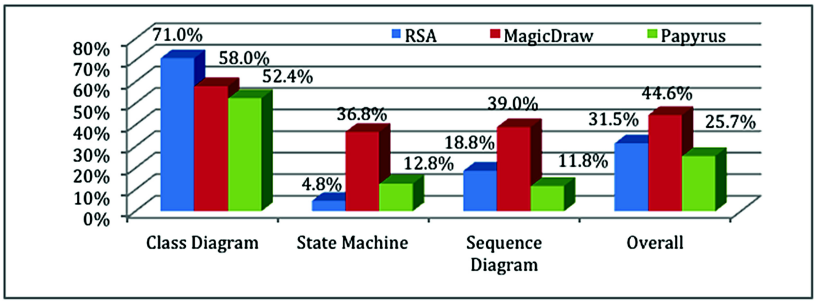

Fig. 1. Average percentage of completed tasks in the three tools

RSA is significantly better than Papyrus in this regard. In terms of completeness for modeling state machines and sequence diagrams, there is no significant difference between RSA-Papyrus. However, MagicDraw is significantly better than RSA and Papyrus for modeling state machines and sequence diagram. We can also observe from Fig. 1 that overall MagicDraw achieved 13% and 20% higher scores for completeness as compared to RSA and Papyrus respectively. Papyrus scored least among all the selected tools. Therefore, among the null hypotheses corresponding to RQ1.1, H1.1.2 holds true whereas H1.1.1 and H1.1.3 are false.

4.2 Effort Required for Modeling

To evaluate the effort required for modeling we measured the time and number of clicks required for modeling. To measure the time we compared the time to draw for each diagram in different modeling tools. Similarly for measuring the number of clicks we compared the number of clicks required for modeling each diagram in different modeling tools. The statistical results of the tests for the time and number of clicks required for modeling are provided in Table 3. We apply the non-parametric Mann-Whitney U-test for all cases while comparing the tools with reference to time and number of clicks required for modeling except in case of state machine. The data sample for state machine has normal distribution; therefore, we apply parametric T-test to compare the time and clicks required for modeling the state machine.

The results of statistical tests corresponding to time required for modeling (Table 3) show that there is no significant different between RSA-MagicDraw and MagicDraw-Papyrus in terms of time required for modeling class diagram, state machine diagram, and sequence diagram. However, in case of RSA-Papyrus results show a significant difference in terms of time required for modeling class diagram and state machine diagram. Participants took significantly more time for modeling class diagram and state machine diagram in Papyrus as compared to RSA. Similarly the results of statistical tests corresponding to number of clicks required for modeling (Table 3) bring us to the same conclusion that participants required significantly more modeling effort for modeling class diagram and state machine diagram in Papyrus as compared to RSA. Among the null hypotheses corresponding to RQ1.2, H1.2.1 and H1.2.3 hold true whereas H1.2.2 is false in case of modeling class diagram and state machine diagram. For all other cases our hypotheses corresponding to RQ1.2 are true.

4.3 Learnability

The data sample for learnability has skewed distribution; therefore, we apply non-parametric Mann-Whitney U-test to compare the selected tools in reference to learnability. The results of statistical tests corresponding to learnability (Table 3) show that there is a significant difference between RSA-MagicDraw, RSA-Papyrus, and MagicDraw-Papyrus in terms of learnability. MagicDraw is significantly better than RSA and Papyrus whereas RSA is significantly better than Papyrus. All the null hypotheses corresponding to RQ2, i.e., H2.1, H2.2, and H2.3 are false. The average values of learnability for each tool on a scale of five are shown in Fig. 2. As one can see from Fig. 2 learnability of MagicDraw was 31% and 14% higher than Papyrus and RSA respectively.

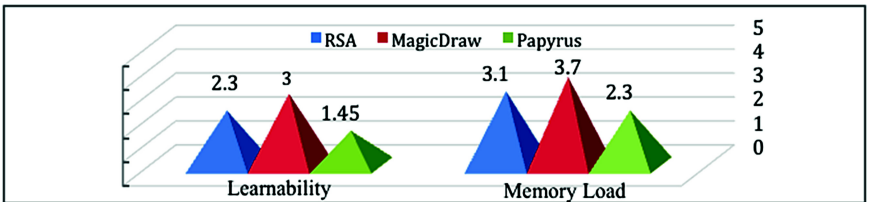

Fig. 2. Average values of learnability and memory load for the three tools

4.4 Memory Load

Our data sample for memory load is skewed, therefore; we apply Mann-Whitney U-test to check the truthfulness of hypotheses and Vargha and Delaney's A12 to find direction and magnitude of difference between two tools. The results of statistical tests for the memory load measure are provided in Table 3. The results of statistical tests suggest that there is no significant difference between RSA-MagicDraw and RSA–Papyrus in terms of memory load measure. However, in comparison of Magic-Draw-Papyrus there is significant difference, MagicDraw is significantly better than RSA in terms of memory load. Among the null hypotheses corresponding to RQ3, H31 and H32 are true whereas H33 is false. The average scores of memory load for each tool on a scale of five is provided in Fig. 2. It can be observed from Fig. 2 that MagicDraw scored 28% higher than Papyrus and 12% higher than RSA Papyrus is the least scoring tool again just like in case of other factors.

5 Discussion

Based on the results provided in Section 4, MagicDraw performed significantly better than the other tools in terms of completeness, learnability, and memory load. Papyrus has the least score in terms of these factors among the three tools. In addition to this we also observed that class diagram has highest score against completeness (i.e., 71%), state machine has a score of 37%, and sequence diagram has only 39% completeness. This can be explained because class diagrams have the least complex modeling elements (e.g., when compared to orthogonal regions in a state machine or combined fragments in a sequence diagram) and is the most widely used of the UML diagrams [7].

Results show that MagicDraw has better learnability as compared to other two tools, which justify why overall MagicDraw is the leading scorer with respect to completeness. We also observed that the models of state machines were the least completed by participants. During the data collection we observed that majority of the participants committed some common mistakes, e.g., they added constraints either using a UML note or in a comment. Similarly a number of students added the *do activity* as a note in state machine. Most likely it is because of the fact that the process of adding such modeling elements, e.g., modeling *do activity* is complex in the modeling tools and require the underlying knowledge of UML meta model.

6 Threats to Validity

In this section we discuss the threats to validity of our experiment in accordance with the guidelines in [13]. The internal threats to validity exist when experiment results

are affected by external factors that are not controlled by the researchers. In our case we used different participants for different treatments of independent variable tools to avoid learning and fatigue effect in case of multiple runs of the experiment. Another internal threat can be poorly designed questionnaire form. To address this we asked authors of the paper, other than the author who developed the questionnaire form, to review the questionnaire form. To avoid the biasness in selection of participants, we formed groups using principles of blocking and balancing. We formed groups using university grading policy and experience with variety of modeling tools. The lack of motivation of participants can be another threat to internal validity. To keep them motivated we assured them that this activity would be treated as a credited course assignment. Another threat is due to the fact that we have not selected all UML diagrams. In practice it is very difficult to cover all the thirteen UML diagrams in a single experiment due to limited resources and fatigue of the participants. Therefore, we have selected the most commonly used UML diagrams (according to [7]): UML Class diagram, UML State Machine, and UML Sequence Diagram.

The external threats to validity are due to settings that hinder the generalization of our results of experiment to industrial practices. The participants of our experiment are students who are not professional software engineers. It is a common practice in empirical software engineering to use students in experiments. This is primarily due to the unavailability of industry professionals for long duration that is required by the experiments. In our experiment, we selected the students with suitable educational background (Section 3.2), and gave them a training session so that they can be treated as representatives of professional designers.

7 Conclusion

MDSE has improved the productivity and quality of software products. UML is a standard modeling language and is widely used in industry to apply MDSE. There are a large number of modeling tools available in the market supporting UML, ranging from open-source tools to commercial tools. To apply UML in practice we need to make a critical decision about the selection of an appropriate tool for modeling. Not all end users are in a position to make such a decision. This paper presented an empirical study to evaluate the productivity of three of the well-known modeling tools: IBM Rational Software Architect (RSA), MagicDraw, and Papyrus. For this purpose we compared these tools in terms of modeling effort required for correctly completing a task, learnability, and memory load required for the engineer to complete a task. The results of our experiment suggest that MagicDraw performed significantly better in terms of learnability, memory load, and completeness of state machine and sequence diagram. MagicDraw outperformed IBM RSA and Papyrus in terms of completeness of state machines and sequence diagram, learnability, and memory load. In terms of time and number of clicks, IBM RSA was significantly better while modeling class diagrams and state machines when compared to Papyrus, however no single tool outperformed others in all the modeling tasks.

Acknowledgement. This work was supported by ICT R&D Fund, Pakistan under the project ICTRDF/MBTToolset/2013. Muhammad Zohaib Iqbal was partly supported by National Research Fund, Luxembourg (FNR/P10/03).

References

1. Iqbal, M.Z., Ali, S., Yue, T., Briand, L.: Applying UML/MARTE on industrial projects: challenges, experiences, and guidelines. Software & Systems Modeling, 1–19 (2014)
2. Khaled, L.: A comparison between UML tools. In: Second International Conference on Environmental and Computer Science, pp. 111–114. IEEE (2009)
3. Eichelberger, H., Eldogan, Y., Schmid, K.: A Comprehensive Survey of UML Compliance in Current Modelling Tools. Software Engineering **143**, 39–50 (2009)
4. Heena, R.: A comparative study of UML tools. In: Proceedings of 11th International Conference on Advances in Computing and Artificial Intelligence, pp. 1–4 (2011)
5. Bellon, S., Koschke, R., Antoniol, G., Krinke, J., Merlo, E.: Comparison and evaluation of clone detection tools. IEEE Transactions on Software Engineering **33**, 577–591 (2007)
6. Ali, S., Yue, T., Briand, L.: Assessing quality and effort of applying aspect state machines for robustness testing: a controlled experiment. In: IEEE 6th International Conference on Software Testing, Verification and Validation, pp. 212–221. IEEE (2013)
7. Dias Neto, A.C., Subramanyan, R., Vieira, M., Travassos, G.H.: A survey on model-based testing approaches: a systematic review. In: Proceedings of the 1st ACM International Workshop on Empirical Assessment of Software Engineering Languages and Technologies: Held in Conjunction with ASE Conference, pp. 31–36. ACM (2007)
8. Sengoz, Y.S., Jawawi, A., Deris, S.B.: A Comparison UML Tools for Real-time Software Modeling (2009)
9. Rani, T., Garg, S.: Comparison of different UML tool: Tool approach. International Journal Of Engineering And Computer Science **2**, 1900–1908 (2013)
10. Bobkowska, A.E., Reszke, K.: Usability of UML modeling tools. In: Proceedings of the Conference on Software Engineering: Evolution and Emerging Technologies, pp. 75–86 (2005)
11. Smith, H.H.: On tool selection for illustrating the use of UML in system development. Journal of Computing Sciences in Colleges **19**, 53–63 (2004)
12. Ali, S., Iqbal, M.Z., Arcuri, A., Briand, L.: A search-based OCL constraint solver for model-based test data generation. In: 11th International Conference on Quality Software (QSIC), pp. 41–50. IEEE (2011)
13. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in software engineering. Springer (2012)
14. Usman, M., Iqbal, M.Z., Khan, M.U.: A model-driven approach to generate mobile applications for multiple platforms. In: Asia-Pacific Software Engineering Conference (2014)
15. Jilani, A.A., Iqbal, M.Z., Khan, M.U.: A search based test data generation approach for model transformations. In: Di Ruscio, D., Varró, D. (eds.) ICMT 2014. LNCS, vol. 8568, pp. 17–24. Springer, Heidelberg (2014)
16. Khan, M.U., Iqbal, M.Z., Ali, S.: A Heuristic-based approach to refactor crosscutting behaviors in UML state machines. In: International Conference on Software Maintenance and Evolution. IEEE (2014)
17. Eriksson, H.-E., Penker, M., Lyons, B., Fado, D.: UML 2 toolkit. John Wiley & Sons (2003)
18. Binder, R.V.: Testing object-oriented systems: models, patterns, and tools. Addison-Wesley Professional (2000)
19. Safdar, S.A., Iqbal, M.Z., Khan, M.U.: Empirical Evaluation of Productivity of Software Engineer in UML Modeling Tools- A Controlled Experiment. Technical report# 3515. Software Quality Engineering & Testing (QUEST) Lab (2015)
20. Sheskin, D.J.: Handbook of parametric and nonparametric statistical procedures. Chapman & Hall/CRC (2007)