A Study Design Template for Identifying Usability Issues in Graphical Modeling Tools

Jakob Pietron[1], Alexander Raschke[1], Michael Stegmaier[1], Matthias Tichy[1], and Enrico Rukzio[2]

[1] Institute of Software Engineering and Programming Languages
[2] Institute of Media Informatics,
Ulm University, 89081 Ulm, Germany
{jakob.pietron, alexander.raschke, michael-1.stegmaier, matthias.tichy, enrico.rukzio}@uni-ulm.de

Abstract. Model-driven engineering aims at increasing the productivity of software engineering and the quality of the software. These positive results have been confirmed in several empirical studies. However, those studies also report that usability of model-driven engineering tools is generally considered to be poor. This is also the prevalent opinion on usability expressed in discussions in academica as well as in our collaborations with practitioners. Unfortunately, there are scarcely any empirical studies on identifying usability issues nor papers reporting on systematically evaluated usability improvements in model-driven engineering. In this paper, we present a study design template for identifying usability issues specifically in graphical editors. This template is grounded in usability research as well as empirical research methods. We illustrate the proposed study design on the example of identifying usability issues in a graphical editor for developing state machines.

Keywords: Usability · Graphical Modeling Tools · Model-driven Engineering · Study Design.

1 Motivation & Problem Statement

The abstraction introduced by model-driven engineering (MDE) aims for building complex systems with high quality in a shorter time. Unfortunately, MDE is still not widely used in industry due to several drawbacks [8]. Condori-Fernández et al. [10] even conclude that most of these drawbacks share the same problem: a lack of usability in the MDE tools. It seems that MDE tool developers are focused on providing frameworks that make it easy to develop domain-specific languages while end-user usability seems to be neglected.

A common argument for less focus on usability is that these tools are only used by experts. This aspect should not count because better usability can also improve the efficiency of experts.

Another problem is that common usability guidelines are not easy to apply, since MDE tools are more than just a means of drawing models. MDE tools

should support an analyst in developing a system [21]. Therefore, our research focus is on analyzing and improving the usability of MDE tools.

Before the usability of MDE tools can be improved, the actual usability issues must be identified. In this paper, we propose a study design template to systematically evaluate the usability of graphical modeling tools. The aim of this study design template is to identify deficits in usability by generating qualitative data instead of quantitative data. While quantitative data, as collected by e. g. Condori-Fernández et al. [10], can indicate problems, make them measurable and enable the comparison of solutions, the actual problems are not identified.

Since we are interested in the real *causes* of poor usability, we propose a qualitative study design. In our study design template we focus on the creation and modification of graphical representations of models. All graphical modeling tools have this functionality in common whereas other functionality such as debugging or model checking is editor and language specific. We suggest performing the evaluations in terms of efficiency, effectiveness and satisfaction, the main aspects of usability as proposed by ISO 9241-11 [13]. Our study design template is meant to form the basis for a family of experiments as advocated by Basili [6] and as such should enable a simple replication of collected data.

We suggest a think-aloud observation after which participants are interviewed and asked to fill in questionnaires. The questionnaires serve to collect experience level, demographic data, and contains questions for the calculation of the System Usability Score (SUS) [9] and the Technology Affinity (TA-EG) [15]. To enable a qualitative analysis of the collected data, coding techniques are used to annotate captured screen recordings.

After discussing the related work in Section 2, we describe our study design template in Section 3 and finally conclude this paper in Section 4.

2 Related Work

There is a lot of literature discussing good practices in conducting usability studies in general (see e. g. [17] or the literature survey in [12]). In all this scientific work different methods for measuring usability or identifying usability problems are introduced and discussed. However, the composition of these methods to a result-oriented study design is often left to the reader. As proposed by Basili [6], especially for the repeatability of experiments guidelines or frameworks are indispensable. In [1], for example, such a framework for measuring the usability of websites is proposed. Similarly, [2] introduces a more general framework, but based on very low-level interactions and therefore difficult to apply in a given context. Moreover, both works aim at usability measurement, not at identifying concrete usability problems.

In the field of modeling tools, which in this context also includes UML tools or domain-specific languages (DSL), the following three different areas can be distinguished:

The first one includes work that focuses on the usability of modeling languages themselves (for an overview, see [20]), but not the usability of the used

tools. In the study conducted by Cuenca et al. [11] similar methods are used as we propose (System usability scale (SUS), observations, and questionnaires, see below), but only the usability of (textual) DSLs is measured and not the usability of the used tools. This also happens in [5] where the authors present "a way to systematize the evaluation process" of DSLs. They introduce a generic process for evaluating DSLs as a user interface of a developer using a DSL. Poltronieri et al. go one step further and define a more precise framework Usa-DSL [19] for this activity in order to carry out replicated usability studies of DSLs. Both papers try to evaluate the DSL itself, but not the (graphical) tools for working with (graphical) DSLs as we do.

In the second area, the usability of tools is considered, but with the focus on tool selection. For example, the work of Safdar et al. [24] is about comparing the usability of different tools in several diagram types. Rouly et al. [23] describe a method for understanding usability of Visual Integrated Development Environments (IDE) used by non-programmers. The considered tools are mostly used for interactive editing of graphical models, yet not necessarily models in the sense of MDE. The usability is measured by analyzing their interface characteristics according to a proposed model. The main goal of this work is the comparison of the tools' usability in an abstract way.

The third area, which is also addressed by us, considers the usability of modeling tools per se. In [7] the authors conduct an experiment to measure the usability of six UML modeling tools. They captured the time each of the 58 participants needed to fulfill several tasks under the assumption, that "the time of performing tasks is one of the usability indicators". The results were confirmed by an analysis with GOMS [14]. GOMS is a formal usability method, that tries to measure the usability of an interface by decomposing typical user tasks into simple basic actions. For each task, the needed time based on estimations per basic user action is calculated and valuated. Besides this simple time comparison, the authors collected participants' comments and suggestions, but not systematically. Interestingly, most of the reported problems obviously have not been improved in the last 13 years.

In [10] Condori-Fernández et al. introduce an evaluation model for identifying the usability of MDE tools by measuring the completeness of user performed tasks in relation to the number of steps needed to finish them. Their framework is defined in a very abstract way by an evaluation model and a generic process. When applying the framework to a tool, the authors propose exemplary methods to be used. Concrete usability problems are discovered by classification of observations against ergonomic criteria. In contrast, our proposed qualitative study design template, which is explained in detail in the following sections, allows usability problems to be discovered on a fine-grained level.

3 Study Design Template

Our study design template can be seen as a generic guideline for how to conduct a case study to identify fundamental usability-related problems of graphical

modeling tools. The study design template focuses on usability problems that occur while users are creating and editing a graphical representation of a model with the functionality provided by a specific modeling tool and not usability problems that are related to a particular domain specific language (DSL).

Yin defines a case study as "*an empirical inquiry that investigates a contemporary phenomenon within its real-life context*" [26]. Furthermore, we can define the type of case study more precisely: it should be an exploratory and explanatory single-case case study. Exploratory to identify the actual usability problems in graphical modeling tools in general and the parts, features, and components of an editor that are responsible for the usability problems in particular. By investigating the reasons why these problems occur the case study additionally becomes explanatory.

As required by Yin [26], in this section, we describe the requirements to the objects of study (editor and graphical modeling language), how tasks should be designed, how to find the right participants, and how to analyze the collected data. For each aspect we discuss the requirements, introduce the solution suggested by our study design template and illustrate it by an example. The example given is based on a study we conducted to evaluate the usability of the graphical modeling tool Yakindu Statechart Tools.

3.1 Objects of Study

The choice of the right editor for a problem discovery study depends on the used graphical language. The chosen graphical language should offer the possibility to define real-world tasks with different levels of complexity. The chosen language should be well known by participants to have no negative effect on the internal validity.

If there is no specific DSL predefined, we suggest state machines as the graphical language for our study design template; state machines are well known to students as well as developers. Therefore, participants can be recruited from a university as well as industrial context. Participants do not have to learn a new graphical language. State machines support a wide range of complexity. It is possible to create very simple state machines but also very complex ones. This makes it easy to create different kinds of tasks for the study that support the participants' learning curve. Furthermore, you can choose from a wide range of different graphical modeling editors that support state machines.

On the other hand, if the editor to be evaluated supports exactly one graphical modeling language and no real users are available for participating in the study, the participants, e.g., students, require a training for that DSL. Regardless of whether real users or just test users participate, for later analysis, it is important to distinguish between problems related to the tool and problems related to the used language.

Example The Eclipse community provides with Eclipse Modeling Framework (EMF), Graphical Editing Framework (GEF), and Graphical Modeling

Framework (GMF) a rich toolbox to create model-driven DSLs and corresponding graphical tools. Many graphical tools such as Papyrus[3], Graphiti[4], and Yakindu Statechart Tools[5] are based on these frameworks. We choose Yakindu as an editor based on GEF to be evaluated. We prefer Yakindu for the following two reasons: first, it supports state machines (see above). Second, it is the tool which focuses the most on graphical editing. It hides a lot of complexity by allowing the user to manipulate and create elements directly in the graphical view. In contrast to, e.g., Papyrus, no dialog windows must be filled in. Most interactions take place inside the graphical view. This in turn supports the internal validity of a study.

3.2 Tasks

To discover usability problems participants should complete tasks with the chosen tool. The tasks to develop should cover a wide range of scenarios and functionality of the chosen graphical DSL and modeling tool. Furthermore, the tasks should become more and more complex in order to support the users' learning curve [17]. The complexity of a task depends on the number of objects, connections and used language concepts. The more objects have to be managed, the more it becomes difficult to keep the modeled diagram readable.

As mentioned before, our study design template focuses on the interaction with the graphical representation of a model which is a diagram in most cases. Therefore, we suggest to define tasks that let participants recreate a given graphically modeled system, e.g., handed out on the printed task description, or edit and refactor a prepared diagram. A participant's result does not have to look exactly the same as in the task description, but it must be functionally equivalent. Setting the task to just create a diagram similar to the one shown on a screenshot ensures that all observed problems are related to the editor and not influenced by a problem in understanding the task or textual system description. Overall, this supports the internal validity.

Besides just creating or editing a diagram, the layout of the diagram created should be readable and comprehensible. We provide a set of rules with the intention to make the participants change a default layouted diagram to improve its readability and comprehensibility. Some of our rules are adopted from the work of Purchase [22]:

- Objects do not overlap
- Edges do not overlap and have a distinguishable margin in between [22]
- Prevent crossing edges [22]
- A label has a clear relation to its edge
- Prevent bends in edge's path [22]
- All labels, names, titles, and descriptions are displayed completely without abridging points

[3] https://www.eclipse.org/papyrus/
[4] https://www.eclipse.org/graphiti/
[5] https://www.itemis.com/en/yakindu/state-machine/

The defined rules are a minimum set of rules that should be fulfilled by a participant's diagram to complete a task. This enforces users to adapt the layout of their diagram by using the tools provided by the evaluated editor. These rules can be extended depending on the chosen graphical modeling language and evaluated editor.

Example A study can consist of different state machine tasks: state machines should be created from scratch, but also existing ones should be edited, and refactored. The tasks should cover a wide range of state machine functionality such as hierarchy, nested states, parallel states, transition with guards in different levels of complexity. Simple state machines consist of only a few connected states without any hierarchy or parallelism. Complex state machines can consist of more than 20 states with up to 50 transitions, hierarchy and parallelism.

All participants receive a prepared Yakindu workspace (following the example from the previous subsection), and for each task, a printed screenshot of a state machine. This state machine has to be previously modeled by the researchers, with the tool of which the usability is to be evaluated. All required events, variables, and if necessary incomplete state machines are prepared a priori in the workspace.

3.3 Participants

At best, real users of the tool can participate in a study. In most cases, however, this will not be the case. Therefore, in this chapter we describe the requirements that the participants must fulfill in order to meet the characteristics of the real users as much as possible.

We assume that users of graphical modeling tools are expert users. Expert users (in contrast to casual users) use a tool regularly for a long period of time. In order to achieve a sufficient external validity, the recruited participants should be at least well experienced in working with computers and software in general, and with graphical modeling tools in particular. The graphical modeling language must also be well known, if not, participants should complete a training beforehand. It is not required that participants have prior experience with the specific editor to be evaluated but they should have experience with any graphical modeling tool for at least 6 months to increase internal validity. To check whether participants fulfill all requirements, the TA-EG questionnaire [15] should be used. Additionally, in a questionnaire participants should be explicitly asked for experience in graphical modeling tools, see following Subsection 3.4.

The TA-EG questionnaire (original German title: Technikaffintät – Elektronische Geräte, translated: Technology Affinity – Electronic Devices) can be used to measure affinity to technical devices like computers, mobile phones, or navigation systems [15]. TA-EG consists of 19 Likert-scale questions grouped into the four categories *enthusiasm*, *competence*, *negative attitudes*, and *positive attitudes*. By using TA-EG, we assess the characteristics of our participants regarding each category. In this way, we can ensure that the participants have a positive attitude towards technology, which we assume corresponds to that of expert users in the context of modeling tools.

Nielsen and Landauer [18] describe the number of participants that are required for problem discovery studies as depending on two factors. First, it depends on the percentage of all usability problems that should at least be found. Second, it should take into consideration the probability of a single problem being found by a participant. Across eleven usability studies, Nielsen and Landauer found the average probability of a problem being found ranges from 0.16 to 0.60 with an average of 0.31. Following the authors' formulation, to detect 90 % of all problems, we require at least nine participants with a problem detection probability of 0.31 (average case). To discover 90 % of all problems with a problem detection probability of only 0.16 (worst case), we require at least 15 participants. Additionally, we require at least one informal participant for a pilot test to fix possible errors in the concrete study design.

3.4 Data Sources

Our study design template benefits from multiple measures that generate qualitative as well as quantitative data: think-aloud observation, a questionnaire, and a semi-structured interview. The three methodologies should be conducted one after the other as listed above.

Using several data sources limits the effects of a possible wrong interpretation of one single data source. Triangulation can be used to increase the validity of observed problems [26].

During the think-aloud observation the participants have to use the system while continuously thinking out loud, and being observed by an experimenter. Thinking out loud means that participants have to verbalize their thoughts, describe what they actually expect and what happens instead. During an observation session, audio and video are recorded. It should be noted that users might give false impressions or own theories of the cause of usability problems. The experimenter should focus on what users actually do instead of what users say they do. For example, a participant might criticize a missing button even though the participant just has not seen it. The real problem is therefore the visibility of the button and not its absence. Later analysis is needed to abstract the observed problems from the participants and identify the underlying usability problems. On the other hand, users' comments on user interface elements, what they like and do not like, can be useful input for later improvements [17].

Directly after finishing the think-aloud observation, participants are asked to fill in a questionnaire. The questionnaire consists of three parts: the System Usability Score (SUS) for measuring the usability of the evaluated tool by a quantitative score, TA-EG questionnaire, and questions about previous experience and demographic data.

SUS [9] is a well established tool to measure usability with a quantitative scale. SUS is used to get a first impression of the usability of the evaluated tool. Furthermore, this score can be compared to possible improvements in the future or other evaluated tools. Our study design contains SUS because it consists of only ten Likert-scale questions. After a possible long lasting think-aloud observation, participants do not want to fill in an extensive usability questionnaire.

Although SUS is a short questionnaire, the resulting score, a value between 0 and 100, is meaningful and valid [4]. The numerical SUS score is not a percentage, despite its appearance. To interpret the numerical score, we use adjective and grade rating scales developed by Bangor et al. [3]. Their work is based on analysis of 1,000 SUS surveys. The average score of the examined SUS-surveys is 68. For example, this numerical SUS score corresponds to *ok* on the adjective rating scale and to *D* on the grade scale.

As introduced before in Subsection 3.2, the TA-EG questionnaire and asking for prior experience are used to validate the sample against the previously defined requirements. Beside just asking for prior experience in general, we recommend to ask explicitly for graphical modeling tools that participants are experienced with and for how long they already use them. This data helps to identify a possible bias that participants might have.

Finally, a semi-structured interview should be used to explore what the users have in mind when working with the editor and to get an understanding of interesting observations during the think-aloud phase [16]. Each participant has to answer pre-determined questions about the overall experience, what was good, and what was bad when they worked with an editor. Aside from the pre-determined questions, some of the interview questions should be based on the observations during the think-aloud phase to get a deeper understanding of the users' behavior in specific situations, e.g., error situations.

Example The researcher noted a participant had problems clicking at a specific node. In the course of the semi-structured interview, the researcher may ask the participant to explain that situation in her or his own words, explain the actual goal and what happened instead. The researcher could also ask for possible improvements.

3.5 Qualitative Analysis

Our study design mainly generates qualitative data. Coding is our proposed way to analyze and structure this data. A code is a short phrase or sentence. It is assigned to a text phrase of the transcribed interviews or video snippet recorded during the think-aloud observation. The code should be the essence or meaning of the coded data [25].

As initially defined, our study design addresses discovery of problems and the context in which these problems occur. We suggest the *open coding* technique to identify problems in the collected data. Open coding breaks down the data into first provisional, comparative codes [25]. Afterwards, we suggest to build up a category system with the focus on affected elements and performed actions by users. The category system should emerge from the coded data.

Example A problem is observed when a participant tries to click at a small connection with the intention to change its position. Instead of clicking at the connection, the participant clicks at the underlying box and moves the box instead. One way to code the described problem might be *connection* (category context) and *click hit* (performed action).

4 Conclusion

Although MDE has a vital research community, its ideas and results are still not well established in the field. One reason might be the poor usability of provided tools and/or provided frameworks to build tools. While there are several studies that quantitatively measure the usability of MDE tools, we try to discover concrete problems in order to work on their improvement.

In this paper, we introduce a study design template that allows for conducting qualitative usability studies more easily. We describe the different aspects of such a study in detail (objects, tasks, participants, data sources, and qualitative analysis) and discuss our suggested methods.

This more abstract description is supplemented by an ongoing example in which the usability of the GEF-based tool Yakindu was examined during the creation or modification of statecharts. We have actually conducted this study and are currently working to fully evaluate the results.

References

1. Al-Wabil, A., Al-Khalifa, H.: A framework for integrating usability evaluations methods: The Mawhiba web portal case study. In: 2009 International Conference on the Current Trends in Information Technology (CTIT). pp. 1–6 (Dec 2009)
2. Andre, T.S., Hartson, H.R., Belz, S.M., McCreary, F.A.: The user action framework: a reliable foundation for usability engineering support tools. International Journal of Human-Computer Studies **54**(1), 107–136 (2001)
3. Bangor, A., Kortum, P., Miller, J.: Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale **4**(3), 114–123 (2009)
4. Bangor, A., Kortum, P.T., Miller, J.T.: An Empirical Evaluation of the System Usability Scale **24**(6), 574–594 (2008)
5. Barišic, A., Amaral, V., Goulão, M., Barroca, B.: Evaluating the Usability of Domain-Specific Languages. In: Software Design and Development: Concepts, Methodologies, Tools, and Applications, pp. 2120–2141. IGI Global, Hershey, PA, USA (2014)
6. Basili, V.R., Shull, F., Lanubile, F.: Building knowledge through families of experiments. IEEE Transactions on Software Engineering **25**(4), 456–473 (Jul 1999)
7. Bobkowska, A., Reszke, K.: Usability of UML Modeling Tools. In: Proceedings of the 2005 Conference on Software Engineering: Evolution and Emerging Technologies. pp. 75–86. IOS Press, Amsterdam, The Netherlands (2005)
8. Bordeleau, F., Liebel, G., Raschke, A., Stieglbauer, G., Tichy, M.: Challenges and research directions for successfully applying MBE tools in practice. In: Proceedings of MODELS 2017 Satellite Event: Workshops (ModComp, ME, EXE, COMMitMDE, MRT, MULTI, GEMOC, MoDeVVa, MDETools, FlexMDE, MDEbug), Posters, Doctoral Symposium, Educator Symposium, ACM Student Research Competition, and Tools and Demonstrations. CEUR Workshop Proceedings, vol. 2019, pp. 338–343. CEUR-WS.org (2017)
9. Brooke, J.: SUS: A "quick and dirty" usability scale. In: Usability Evaluation in Industry. Taylor and Francis (1986)
10. Condori-Fernández, N., Panach, J.I., Baars, A.I., Vos, T., Pastor, Ó.: An empirical approach for evaluating the usability of model-driven tools. Science of Computer Programming **78**(11), 2245–2258 (2013)

11. Cuenca, F., Bergh, J.V.d., Luyten, K., Coninx, K.: A User Study for Comparing the Programming Efficiency of Modifying Executable Multimodal Interaction Descriptions: A Domain-specific Language Versus Equivalent Event-callback Code. In: Proceedings of the 6th Workshop on Evaluation and Usability of Programming Languages and Tools. pp. 31–38. PLATEAU 2015, ACM, New York, NY, USA (2015)
12. Hornbæk, K.: Current practice in measuring usability: Challenges to usability studies and research. International Journal of Human-Computer Studies **64**(2), 79–102 (2006)
13. ISO: ISO 9241-11:1998 Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability. Tech. rep., International Organization for Standardization (1998)
14. John, B.E., Kieras, D.E.: The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast. ACM Trans. Comput.-Hum. Interact. **3**(4), 320–351 (Dec 1996)
15. Karrer, K., Glaser, C., Clemens, C., Bruder, C.: Technikaffinität erfassen–der Fragebogen TA-EG. pp. 196–201. No. 8 in Der Mensch im Mittelpunkt technischer Systeme (2009)
16. Lazar, J., Feng, J.H., Hochheiser, H.: Research Methods in Human-Computer Interaction. Wiley (2010)
17. Nielsen, J.: Usability Engineering. Academic Press (1993)
18. Nielsen, J., Landauer, T.K.: A mathematical model of the finding of usability problems. pp. 206–213. ACM Press (1993)
19. Poltronieri, I., Zorzo, A.F., Bernardino, M., de Borba Campos, M.: Usa-dsl: Usability evaluation framework for domain-specific languages. In: Proceedings of the 33rd Annual ACM Symposium on Applied Computing. pp. 2013–2021. SAC '18, ACM, New York, NY, USA (2018)
20. Poltronieri Rodrigues, I., Campos, M.d.B., Zorzo, A.F.: Usability Evaluation of Domain-Specific Languages: A Systematic Literature Review. In: Human-Computer Interaction. User Interface Design, Development and Multimodality, LNCS, vol. 10271, pp. 522–534. Springer (2017)
21. Post, G., Kagan, A.: User requirements for OO CASE tools. Information and Software Technology **43**(8), 509–517 (2001)
22. Purchase, H.: Which aesthetic has the greatest effect on human understanding? In: Graph Drawing. pp. 248–261. Springer (1997)
23. Rouly, J.M., Orbeck, J.D., Syriani, E.: Usability and Suitability Survey of Features in Visual Ides for Non-Programmers. In: Proceedings of the 5th Workshop on Evaluation and Usability of Programming Languages and Tools. pp. 31–42. PLATEAU '14, ACM, New York, NY, USA (2014)
24. Safdar, S.A., Iqbal, M.Z., Khan, M.U.: Empirical Evaluation of UML Modeling Tools–A Controlled Experiment. In: Modelling foundations and applications, Lecture Notes in Computer Science, vol. 9153, pp. 33–44. Springer, Cham (2015)
25. Saldaña, J.: The Coding Manual for Qualitative Researchers. SAGE, 3rd edn. (2016)
26. Yin, R.K.: Case Study Research: Design and Methods. SAGE, 5th edn. (2014)