2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)

Usability of Development Tools: A CASE-Study

Thomas Weber
fortiss GmbH
Munich, Germany
weber@fortiss.org

Prof. Dr. Alois Zoitl
Johannes Keppler University
Linz, Austria
alois.zoitl@jku.at

Prof. Dr. Heinrich Hußmann
Ludwig-Maximilans-University
Munich, Germany
hussmann@ifi.lmu.de

***Abstract*—To facilitate model-driven development in practice, we need tools that support and empower developers. Many of the tools for it, however, are somewhat lacking with respect to usability, which can act as a major obstacle in adopting a model-driven approach and impede productivity. Efforts to find concrete usability flaws in model-driven tools are still relatively rare, and in the human computer interaction community, usability for highly specialized expert software is also not a prominent topic. In this work we conduct an evaluation using a number of usability evaluations methods on the 4diac IDE as an example of a model-driven engineering tool to see what works and where the problems are. With those different methods we found a considerable number of usability issues of varying severity as well as some indication which methods work in what context. These findings of course help to improve the 4diac IDE specifically but also offer more general insights that may benefit other tools in improving their user experience. Overall our evaluation shows that there is a definite need for better usability and more evaluation of it.**

***Index Terms*—Model-Driven Engineering, CASE Tool, Usability, 4diac**

I. Introduction

While the benefits of model-driven engineering are quite clear in theory, for them to be realized in practice requires good tools that enable novices and experts alike to create, analyze and maintain models that encompass their domain knowledge, the problem space, and the solution. The goals should be to empower developers by giving them flexibility where needed and constraining them when necessary.

For this to be possible, these tools must conform to the established best practices of tool-, interface- and usability engineering. Only by being usable, supportive of their users goals, and by facilitating learning can they fulfil their purpose. However, the unfortunate reality is that tools for MDE are somewhat lacking when it comes to usability [8]. Especially, but not exclusively, tools developed in research and in open source projects are prone to this. This has many unwanted effects like limiting their users' productivity or comprehension of the systems they create, something that is directly contrary to the stated purpose of these tools. It the worst case, poor usability results in users simply not using the tool, which not only means a lot of development effort gone to waste, but also stops users from benefiting from the gains of MDE, even if they are aware of the theoretical benefits [5].

There are of course reasons for this situation: research often requires these tools as a mere means to the end to showcase results so usability is not an issue. Once they transition from pure research prototypes to being used productively, it should be though. Frequently these tools end up as open source projects, which by no means is a bad thing, but has some drawbacks with respect to improving the usability. For one, open source projects have limited resources, as they often rely on volunteers and donations, which are invested more into additional features [9]. The meritocratic governance structure common in open source projects additionally raises the entry barrier for volunteers that want to contribute in less tangible ways, like improving the usability [1], [11], [14]. This, again, is no criticism of the open source format, especially since commercial tools can have the same usability issues, but it shows some of the factors that contribute to the situation of development tools in general and model-driven tooling in particular.

In this work we evaluate a CASE tool for MDE, 4diac (Sect. III), using a variety of well-established methods for empirical usability evaluation (Sect. IV). We hope that our findings, as outlined in Sect. V, help to bring stronger awareness to these issues within the MDE community, give those working on tools some means for further improving their work, and also give some specific pointers to issues that may be pervasive across MDE tools and match up with the findings from other CASE tool evaluations.

II. Related Work

Concrete indication of the usability problems, for commercial and open source software alike, can be found aplenty in anecdotes but also some publications [8].

In the plethory of usability evaluations in research, those with a focus on CASE (computer-aided software engineering) tools are rare though, even more so for tools in MDE. There are certainly examples of individual systematic evaluations, like Vigo et al. [18] who investigated how people use modeling tools using eye-tracking. Based on their results they to give some design guidance but as they point out, the nature of specialized tools reqiures finding some broad common denominator, making it uncertain, how applicable these results are to other domains or tools. Fowler et al. [6] or Teruel et al. [17] conducted an evaluation of a tool for requirements engineering. Similarly, Mealy et al. [10] investigated refactoring tools, two of which were, just like the 4diac IDE we investigated, Eclipse-based. Results in these evaluations were mixed and given the specialized nature of requirements engineering or

978-1-7281-5125-0/19/$31.00 ©2019 IEEE
DOI 10.1109/MODELS-C.2019.00037

refactoring, it is again debatable how generalizable the findings can be. Teruel et al. do point out some key challenges for these evaluations however, namely choosing appropriate test subjects and tasks given the complex nature of CASE tools. This also makes comparison of tools challenging, since individual issues are hardly comparable, so authors like Safdar et al. [15] have to rely on generalization or aggregation, which does not allow extrapolation to other tools.

There is also some work on different combinations of the model-driven world and the usability domain, for example by integrating usability in the modelling process such that the models can be inspected for potential usability issues later in the actual application [4]. The constructive nature of this approach should prevent usability problems from finding their way into the application from the beginning but it has some limitations in scope and unpredicted influences. And while it may be a way to get MDE practicioners enganged with usability in a familiar environment, this work has not reached the maturity of more common usability approaches, and thus was no feasible alternative for us.

III. System Under Test

The following section briefly outlines the design and purpose of the Eclipse 4diac IDE [16] as our test subject. The choice fell onto Eclipse 4diac because of the close association with its development team, but as it is based on the Eclipse IDE and generally similar to other MDE tools, we are convinced that the results of this evaluation can bring value to other tools and projects.

The 4diac IDE is a specialized tool for graphically developing systems with a focus on industry automation. The development process is model-driven, based on the IEC 61499 standard. Applications are constructed from Function Blocks (FBs) which communicate via signals and data ports. There is an extensive library of predefined FBs available as well as the functionality to create new ones. The developer can choose between state machines, structured text or custom implementations for implementing new FBs. FBs can also be an encapsulated composition in a sub-graph of FBs. Next to defining application logic with FBs, developers also assign functionality of FBs to individual devices and controllers in a system and network configuration task. This incurs an overhead in coordinating between those devices but allows developers to easily deploy their application across multiple controllers, as long as they run the 4diac FORTE runtime. The deployment process onto those devices is also supported in the 4diac IDE's interface, as well as debugging and monitoring which allows developers to interactively inspect their application.

The interface is based on the Eclipse IDE and platform, following its conventions and overall design. The individual steps in the application life-cycle, defining the FB network, creating new FBs, configuring the system and assigning functionality to resources, deployment, and debugging, are separated into different views or perspectives (Fig. 1). Most constructive tasks, like building the FB network, can be performed via drag and drop and direct manipulation. This also includes some of the configuration work and setting parameters. Other configuration tasks, programming with structured text or documentation use traditional forms. Errors and warnings also closely follow platform conventions as either console output or via notifications.

There is also a full workflow on recompiling the runtime with new custom FBs that involves external tools, but this was not part of our evaluation.

Since the models and semantics used in the 4diac IDE are constrained by the underlying standard, we do not evaluate them but rather focus on how the 4diac IDE implements the user interface to make the standard accessible to developers. We therefore also did not evaluate the setup phase to get the 4diac IDE to run stand-alone or in conjunction with physical devices.

The version of the 4diac IDE we used during our evaluation was 1.10.0. To perform some of our studies we instrumented the IDE using a plugin that was able to record user interaction with the interface as well as provide instructions (see Sect. IV-G).

IV. Methods

For evaluating the IDE, we relied on a set of methods, common in the evaluation of UIs. The following section gives a brief overview of those methods and their execution.

A. Expert Review

The expert review is probably the most straight forward although unstructured of the methods we used. One or more experts for usability inspect the system and note all aspects that they, with their experience, deem unsatisfactory for potential users. This of course requires the experts to be aware of the users demands, goals, and needs. Likewise, the experts should have at least a basic understanding of the system and what it aims to achieve.

We had a single UX expert review the Eclipse 4diac IDE. It may be noted that this reviewer was generally familiar with software engineering tools, graphical and non graphical and had worked through the Eclipse 4diac tutorial to get familiar with the IDE.

B. Heuristic Evaluation

A more structured approach to experts review is Nielsen's heuristic evaluation [12]. It works along common heuristics for good UI design and provides experts with a questionnaire containing items related to these heuristics. This questionnaire is commonly coupled with specific tasks.

As with the unstructured expert review, it relies on the expertise of the reviewer although an increased number of reviewers can yield adequate coverage [13].

Since reviewers of different backgrounds focus on different aspects, the lack of domain knowledge for UX experts can be alleviated by having different reviewers, with usability expertise and with domain knowledge, where the questionnaire acts as a guidance.

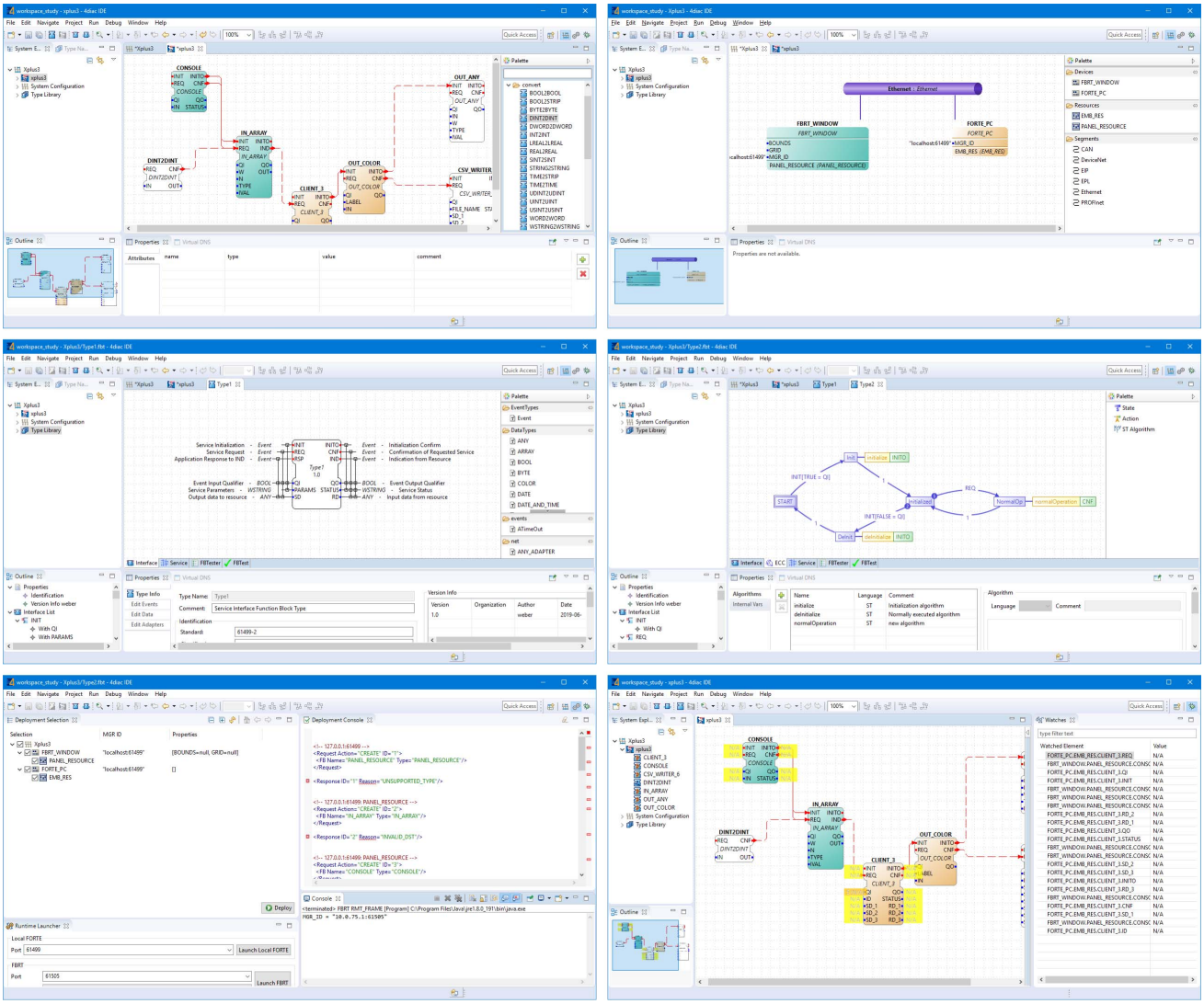

Fig. 1. The different perspectives in the 4diac IDE for (top to bottom, left to right): building the application from FBs, configuring the network, defining new FB types, defining FB behavior by state machine, deployment, debugging.

We performed the heuristic evaluation with a total of five evaluators, two of which were from the UX domain, two were users and one was a developer of Eclipse 4diac.

C. Focus Group

A focus group is a form of user involvement in the design or evaluation process. It involves a group of ideally diverse stakeholder, in our example that could be developers, expert and novice users of the IDE and users of machines programmed with it, and maybe managerial staff that receives information from their employees demonstrated via the 4diac IDE. These stakeholders are gathered for a moderated discussion, usually along some guiding questions. It is also possible to use some creativity methods, props, etc.

Focus groups are usually utilized in early phases of product development, since they usually identify user needs and goals in a qualitative format.

We coupled our focus group with an Eclipse 4diac training session which not only made it easy to get people to come together but also made for a diverse group with some complete beginners, some more experienced users and the expert who delivered the tutorial.

The group followed along the development process outlined in the tutorial with print-outs of the most important steps. We asked them to first individually and then collectively give feedback along the presented tasks and with the help of the print-outs. The results we discussed in the group. After collecting the feedback we also asked the participants to provide feedback which aspects they prioritized.

D. Interview

The interview is a commonly known format of a usually one-to-one discussion between interviewer and interviewee. One can differentiate between unstructured interviews, which are more akin to a conversation, structured interviews, which follow a clear, predefined catalog of questions, and semi-structured interviews, which provide a guideline of questions but allow for discussion beyond those questions.

Interviews are particularly suited to delve into specific topics in detail, but, like focus groups, yield mostly qualitative results. Furthermore, interview results are dependent on the expertise of the reviewer.

In our work we used the latter, semi-structured interviews. The focus of our interview questions was on how they currently use the 4diac IDE, what the main challenges in usage are, how they can be overcome, and how general the interviewee deemed these issues and fixes. We also briefly touched on the subject of how suitable graphical programming and MDE were for the specific domain and in general.

E. (Online) Survey

A format equally if not more common than interviews is the survey, nowadays mostly conducted online due to the ease and range of distribution. It asks participants to answer a set of questions regarding the topic of interest. Results can be both quantitative, e.g. via Likert-scales, and qualitative via open text answers. Unfortunately, especially of open questions, participants are reluctant to give extensive textual answers and if they do provide written feedback, it can greatly vary in quality.

The design of surveys is also a challenge, since, for qualitative results, their asynchronous format makes it impossible to delve deeper into topics or ask follow-up questions. For quantitative data, the proper operationalization, format, wording or order of questions can also be highly influential on the results if not carefully designed.

Some of the standardized or conventially accepted questionnaires (e.g. [**?**], [**?**], [2]) can offer guidance but they primarily give only an aggregated score for a user interface. So to find specific, conrete individual issues they are are not necessarily suited.

F. Lab Experiments

The probably most work-intensive format for user research is performing usability evaluations in a lab setting or a similar controlled environment. It requires the user to perform a set of clearly defined tasks while the evaluator observes the user during this task, traditionally via a one-way mirror or by camera, so as not to interfere with the task. Less sophisticated setups are of course also possible but can potentially introduce interfering factors.

The instructions for the participant may vary, based on what is being evaluated. It may range from simply performing a task where completion time or error rate is measured quantitatively up to asking the user to provide constant spoken feedback of activities, perception, and mental-processes during task execution. The latter, called “think-aloud” [3], can yield valuable qualitative insights.

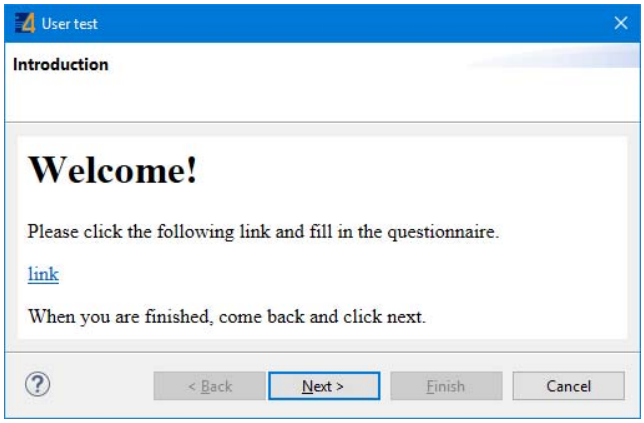

Fig. 2. The instructions provided by our plugin for remote user test participants.

We used the “think-aloud” format and designed the tasks along the development life-cycle in the 4diac IDE, including modeling a system, configuring the system, deployment, and, if necessary, debugging. We roughly aligned the difficulty of our tasks with that of 4diac tutorials. The tasks were given by the examiner, who also assisted when the participant found no solution to the task.

G. Remote Experiments

To mitigate the overhead of getting participants into a lab setting, evaluation can also be conducted remotely. We differentiate between synchronous and asynchronous remote test.

During *syncronous remote testing* the participant communicates via the evaluator via for example Skype, allowing immediate feedback and supervision. With screen-sharing technology, the evaluator can even observe what the participants do in the software although not what happens beyond the screen.

For *asynchronous remote testing*, the participants can choose the time when to perform the study on their own, giving maximum flexibility. Since the evaluator does not communicate with the participant, the prepared instructions must be carefully constructed to minimize the risk of misunderstandings or failure of the experiment. Likewise, there has to be a system in place for either automated observation or recording of the users action and means for the user to provide feedback. This of course means a higher workload in preparing the study but given the flexibility of this approach it may limit the entry barrier making it more likely people will participate.

For our evaluation we instrumented the 4diac IDE with an Eclipse plugin, based on the RCP Test Tool, modified such that it could provide instructions, both in text and image (see Fig. IV-G), and also recorded the users interactions and spoken feedback. The interaction recording happened at the level of UI elements, i.e. the resulting log listed at which point in time

the user clicked what UI element or typed in which input field. This allowed us to accurately play back the actions of a remote participant at a later time and very accurately measure the time on task for the individual tasks and sub-tasks. Between each task the plugin also stored snapshots of the workspace.

This plugin was preinstalled in the version of the 4diac IDE we distributed to our participants.

H. Other Methods

Of course there are plenty other methods for user research. Especially field observations are popular, due to them having users act in a realistic environment. They do, however, pose the same challenges. Like in classic lab testing participant and evaluator need to be coordinated in time and location. It also means that the evaluator observes the participant during their regular work, which, due to industry secrets, privacy etc., can be undesired. Lastly, it also impedes comparability, unless the observed all perform a roughly similar task. Weighing the necessary additional overhead and the potential benefit of field observations, we decided to focus on the methods listed above.

The same applies for more elaborate methods, for example using eye tracking electro-dermal activity or other sensors in the lab experiments which can bring additional insights but require additional overhead which, in our view, was not justified for our work.

In addition, more methods always require participants. Given the limited user group of the Eclipse 4diac IDE, finding participants for the methods we used was already sufficiently challenging.

V. Results

Overall the results of the usability evaluation indicate that there is a lot of potential for improving the 4diac IDE. This section will give a brief overview of the effectiveness of the individual methods. A complete list of 4diac specific findings is available upon request.

A. Expert Review

The expert review was conducted by a single UX expert and found a total of 38 usability issues in the 4diac IDE. This may appear like a relatively large number, but closer inspection revealed an issue: Since, especially in a so specialized domain like MDE, it is somewhat unrealistic that a UX expert is familiar with the minutiae, they will focus on the aspects that they can assess, which can result in superficial feedback.

When differentiating between syntactic issues, i.e. pertaining to visual presentation, or general interaction patterns, and semantic issues, i.e. those affecting the users mental model or system understanding, we saw that the expert found primarily syntactic issues. In fact, of the 38 issues, 36 were syntactic issues, of which 12 were about inconsistent behaviour and 11 cosmetic. Fig. 3 shows the distribution of syntactic and semantic issues for all the methods we used.

This, to us, is indicative of the fact that the reviewers limited knowledge of the MDE process can skew the results towards more superficial issues. This in no way devalues the findings

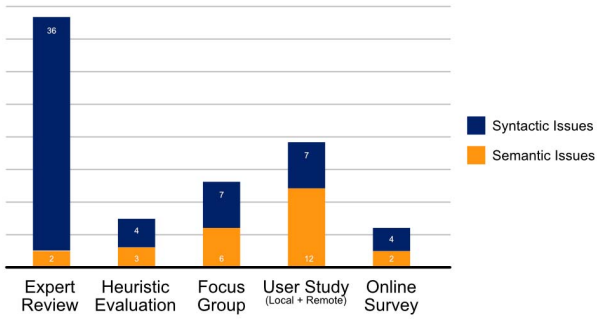

Fig. 3. The proportion of syntactic, i.e. presentation and interaction, and semnatic issues, i.e. those regarding understanding.

of the expert, as they are legitimate issues, but one must be aware that this method is limited in this respect and should not be the only choice for a comprehensive evaluation. It is especially the case, since both users and experts ranked the majority of semantic issues with a higher severity than the syntactic issues.

B. Heuristic Evaluation

The results of our five heuristic evaluators were mixed with overall high scores for the “Match between System and the Real World” and “Consistency and Standards” category. This is unsurprising, given that the 4diac IDE is based on an IEC standard. Likewise, the low score for “Flexibility and Efficiency of Use” can be attributed to some of the rigid necessities of the standard, but also to the many repetitive tasks during development which limit the efficiency.

An interesting observation during the heuristic observation was that even the more experienced users stated that they had no full confidence in giving criticism, due to the perceived lack of expertise. It appears that the complexity of a system like the 4diac IDE is intimidating to evaluators who are not sure whether an issue is due to a fault in the IDE or because of their perceived lack of knowledge. This may be an indicator that for these complex systems an evaluation method without a examiner with some domain knowledge, may not be the ideal choice, but also that the entry barrier in terms of domain knowledge is fairly high for 4diac and probably for similar tools too.

C. Focus Group

Eight people participated in our focus group.

The focus group stands out from the other methods in that the participants focused much less on finding issues with the existing IDE but rather made requests for additional features. These feature requests can be traced back to problems that are currently not addressed in the UI or to the fact that existing features are hard to discover.

The diversity of the group also proved beneficial, yielding interesting discussions between experts and novices. Overall

we did see the strongest participation from the intermediate users though. This may be specific to our group but also shows that some facilitation is necessary to encourage especially the novices. The use of print-outs of the UI for giving feedback did work very well, giving us some additional cues for later analysis that had not been discussed in the group.

D. Interviews

We interviewed five industry users of the 4diac IDE in a semi-structured fashion. The semi-structured interview format worked well, allowing us to ask for existing issues and delve deeper into suggested solutions.

The graphical format of the 4diac IDE was rated as positive. The interviewed experts saw graphical programming as a viable method as long as there was the option to program non-graphically. This is in line with their opinion that the hierarchical nature of many graphical programming methods is beneficial for understanding and communication while not restricting the programmer. This also applies for MDE tools which often take a hierarchical approach, composing larger models from smaller parts. This allows developers to communicate on a high level and work on details all in the same context.

E. Online Survey

After distributing our online survey via the Eclipse 4diac social media accounts and some direct contact, we received ten submissions. This already shows the difficulty of getting a sufficient amount of participants. Given that the most valuable feedback often comes from the free-text answers and not all participants have the time or motivation to write detailed text, the survey depends on the quantity of submissions to get representative quantitative results and a decent number of qualitative feedback.

We did however find high engagement with our participants, with eight answering some of the free-text questions and four giving fairly extensive feedback, which led to some immediate improvements. The quantitative results were very mixed, giving no clear indication where concrete issues might be.

F. Lab Experiment

For the lab experiment we invited seven participants to perform a series of predefined tasks under observation. The overall engagement was high although the participants did not consistently express their thought process. This may be because the usage of a complex tool like the 4diac IDE takes up too much mental capacity.

The participants also relied heavily on the instructions of the evaluator with little to no experimentation or exploration. This was the case even though the tasks we selected were similar to those in a novice tutorial. Given that multiple participants compared the tasks to their own work, it may be that not working on a familiar problem was limiting the participants.

In total, the participants did find 14 issues and expressed some general opinions on the up- and down-sides of tools like the 4diac IDE. The co-location of evaluator and participant seemed to encourage general discussion on tool usability as well as positive feedback.

G. Remote Experiment

We conducted remote experiments with $n = 9$ participants, six of which in a synchronous, three in asynchronous setting.

They performed the same tasks as in the lab experiments, using an instrumented version of the 4diac IDE. The instrumentation, an Eclipse plugin, allowed us to record the actions of the participant in the IDE, allowing us to replay the session and making it easy to extract timing information. For the asynchronous setting, we could also show the instructions for the participants using the plugin.

In total, the participants found 29 issues in the remote experiment including the only one that was ranked as catastrophic severity. Whether the remote setting had an effect on the severity of the findings cannot be said though.

We also compared the remote experiments with the local one to see, whether remote studies are a viable alternative to the more costly local setting. This comparison was with respect to number of found usability issues as well as the participants perception, measured via the NASA TLX [7], and the overall task completion time. Except for the time on a single subtask we did not find any significant differences between the conditions, indicating that remote studies are a viable alternative. As mentioned before though, having participant and evaluator co-located can spark spontaneous discussions, which especially asynchronous remote studies cannot.

H. Other Results

Throughout all methods we found it somewhat challenging to find users as participants. This not because of unwillingness; on the contrary, Eclipse 4diac users were eager to help to improve their tool. The Eclipse 4diac project targets such a specialized domain though. Consequently there is no large, easy to reach community, so we had to rely on personal connections to users, which led to the difficulty in finding participants. This probably is the case for other specialized tools as well, while more general purpose tools may suffer less from it. Furthermore, in a real-world evaluation, there is no need to execute the full range of methods, a selection will suffice. To avoid learning effects and other effects on the result quality, we also chose a new group of participants for almost each method. With a single sub-group of three participants, we conducted both interviews and user tests, which worked out well, so this may be another way to work with a limited number of participants.

Another issue we encountered during multiple methods was the fact that the expertise of participants not only varied greatly but was also distributed very differently. There were tasks that a participant would be highly proficient in, but on other task the participant struggled. By their own reporting, this was the case because they had to use a feature of the 4diac IDE that they, in their daily work, did not use. The 4diac IDE and other complex software development tools, are full of features for

very different tasks. Even when going along with a relatively simple tutorial, we ended up with tasks that were of very different difficulty for different participants. This clearly shows the challenge with defining good, common tasks that yield comparable results. Again, with more general purpose tools this issue might be less prominent.

VI. Discussion

As described in the previous section, our results were somewhat mixed. Some methods, like expert review and experiments with real users worked fairly well, but had some drawbacks like focusing only a subset of issue or being very laborious respectively. A method like the focus group proved a good way to find new feature requests but had limited use in finding existing usability issues. Other methods again, like the online survey were unconclusive given the limited number of participants we could recruit.

Considering the low number of participants, we do not compare the number of issues found per method. While some methods clearly did find more issues, not only is it hard to compare these numbers, given that the issues differ greatly in nature, but there are many more factors that contribute to this number. Instead we focus on the methodological insights we have gained from conducting these methods with software developers and MDE practitioner.

There is not one method that can always be applied but the benefits and drawbacks have to be weighed to choose the right method given the special context of CASE and MDE tools.

The most important factor is the limited number of users, which in turn means limited test participants. This is a considerable impediment for quantitative methods which rely on a large number of participants to get representative results.

Another factor that needs to be taken into account with development tools is the complexity of them. The specialized nature of domain specific tooling adds further to this. A generic usability expert cannot have the detailed domain knowledge that may be necessary to decide whether an issue is due to a real issue or just lack of knowledge. This is a strong argument for user tests, i.e. tests with real users, ideally in real scenarios. As we discussed, they incurr a high overhead and suffer from the aforementioned limitation in particpant numbers.

Since it is possible to build a wide variety of applications using the 4diac IDE, it is also particularly challenging to find good, common tasks for users to perform. The reluctance of participants in the experimental setting to explore and try things beyond the explicit instructions, may be an indicator that they were overwhelmed with these relatively simple tasks. So to find issues, real-world observations may be the better choice. They are, as mentioned, not without their own challenges though.

And then there is the matter, as pointed out in an interview, that software developers can have very particular preferences that may go against convention or consensus, so results can be contradictory, making flexibility very important.

One possible recommendation for development of CASE and MDE tools could be to choose the method depending on the phase in the life cycle: In the very beginning focus groups are viable to elicit feature requests. Since they do not have to comprised entirely of domain experts, they do not suffer as severely from the low users numbers as other methods do. During development of individual features, the feedback of a usability expert can be valuable to prevent obvious fallacies. When a feature has reached a certain maturity, it may be evaluated in small experiments. By focusing the experiments on specific features, users can participate in multiple experiments for different features, which allows building a pool of willing test subjects. Since they are then also familiar with the general experiment process, the overhead may decrease.

Just as the lack of users and therefore test participants is impeding usability evaluations, it also impacts this assessment: we only had a handful of participants for each method and used a selection of evaluation methods. And while this has to be kept in mind with respect to generalizability, our findings are overall consistent with both anecdotal knowledge and the results from the literature.

Likewise we focused on one specific tool in our evaluation. The 4diac IDE cannot be representative of all CASE or even MDE tools. Being Eclipse based, a platform for many other tools, is a benefit though. A note in the heuristic evaluation in fact points out that some usability issues are not specific to the 4diac IDE but an issue of the underlying platform. Additionally, many interaction patterns and design choices from 4diac IDE are present in other MDE and other tools and our methodological results are for the most part independent from the system under test.

VII. Conclusion

In this work we have thoroughly evaluated the usability of the 4diac IDE, a tool and platform for developing industry automation software in a graphical, model-driven fashion. We have used different methods in this usability evaluation with the goal of finding as many issues as possible and seeing which methods work best for what purpose. Overall we found an abundance of usability issues which shows that there is great potential for improving the 4diac IDE. This finding is not specific to just the 4diac IDE but we are convinced our results apply to other graphical and MDE tools as well. For this reason, we have given a list of both the issues we found and possible lessons to learn for the development and improvement of other tools. We have also outlined the strengths and weaknesses of the individual methods in the context of MDE and development tools, as well as some general findings and recommendations regarding usability evaluations for such specialized tools.

And while we clearly see a number of challenges, like finding a sufficient amount of qualified test participants, and designing good experiments, we are convinced that these efforts are worthwhile. The reception of our work to improve the usability of the 4diac IDE was exceptionally positive with Eclipse 4diac users, showing us that there is a demand for good

usability. Participants showed an enthusiasm and gratitude and concurred in our assessment that good usability will be beneficial for productivity and user satisfaction. They also agreed that graphical, model-driven development is absolutely capable of delivering this.

So, we hope that this work inspires other practitioner, researchers, and developers, open source or commercial, to spend some time and effort on improving the usability of the tools they create, such that they can help spread the benefits of MDE.

ACKNOWLEDGMENT

This research was funded by the Bavarian Ministry of Economic Affairs, Regional Development and Energy.

REFERENCES

[1] M. S. Andreasen, H. V. Nielsen, S. O. Schrøder, and J. Stage. Usability in open source software development: opinions and practice. *Information technology and control*, 35(3), 2006.

[2] J. Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[3] K. A. Ericsson and H. A. Simon. *Protocol analysis: Verbal reports as data.* the MIT Press, 1984.

[4] A. Fernandez, S. Abrahão, E. Insfrán, and M. Matera. Usability inspection in model-driven web development: Empirical validation in webml. In A. Moreira, B. Schätz, J. Gray, A. Vallecillo, and P. J. Clarke, editors, *Model-Driven Engineering Languages and Systems - 16th International Conference, MODELS 2013, Miami, FL, USA, September 29 - October 4, 2013. Proceedings*, volume 8107 of *Lecture Notes in Computer Science*, pages 740–756. Springer, 2013.

[5] A. Forward and T. C. Lethbridge. Problems and opportunities for model-centric versus code-centric software development: a survey of software professionals. In J. M. Atlee, R. B. France, G. Georg, A. Moreira, B. Rumpe, S. Völkel, and S. Zschaler, editors, *International Workshop on Modeling in Software Engineering, MiSE 2008, Leipzig, Germany, May 10-11, 2008*, pages 27–32. ACM, 2008.

[6] L. Fowler, J. Armarego, and M. Allen. CASE tools: Constructivism and its application to learning and usability of software engineering tools. *Computer Science Education*, 11(3):261–272, 2001.

[7] S. G. Hart and L. E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52, pages 139–183. Elsevier, 1988.

[8] T. C. Lethbridge. Key properties for comparing modeling languages and tools: Usability, completeness and scalability. In A. Moreira, G. Georg, G. Mussbacher, J. Kienzle, R. B. France, and S. Ali, editors, *Proceedings of the Fourth International Comparing Modeling Approaches Workshop 2013 co-located with the ACM/IEEE 16th International Conference*

[12] J. Nielsen. 10 usability heuristics for user interface design, 1994. https://www.nngroup.com/articles/ten-usability-heuristics/ Retrieved August 11, 2019.

on Model Driven Engineering Languages and Systems (MODELS 2013), Miami, Florida, USA, October 1, 2013., volume 1076 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.

[9] A. L. Masson, D. Lalanne, and T. Amstutz. A usability refactoring process for large-scale open source projects: The ILIAS case study. In G. Mark, S. R. Fussell, C. Lampe, m. c. schraefel, J. P. Hourcade, C. Appert, and D. Wigdor, editors, *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, Denver, CO, USA, May 06-11, 2017, Extended Abstracts.*, pages 1135–1143. ACM, 2017.

[10] E. Mealy, D. A. Carrington, P. A. Strooper, and P. Wyeth. Improving usability of software refactoring tools. In *18th Australian Software Engineering Conference (ASWEC 2007), April 10-13, 2007, Melbourne, Australia*, pages 307–318. IEEE Computer Society, 2007.

[11] R. Z. Moghaddam, M. Twidale, and K. A. Bongen. Open source interface politics: identity, acceptance, trust, and lobbying. In D. S. Tan, S. Amershi, B. Begole, W. A. Kellogg, and M. Tungare, editors, *Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011, Extended Abstracts Volume, Vancouver, BC, Canada, May 7-12, 2011*, pages 1723–1728. ACM, 2011.

[13] J. Nielsen and R. Molich. Heuristic evaluation of user interfaces. In J. C. Chew and J. A. Whiteside, editors, *Conference on Human Factors in Computing Systems, CHI 1990, Seattle, WA, USA, April 1-5, 1990, Proceedings*, pages 249–256. ACM, 1990.

[14] M. Rajanen and N. Iivari. Power, empowerment and open source usability. In B. Begole, J. Kim, K. Inkpen, and W. Woo, editors, *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI 2015, Seoul, Republic of Korea, April 18-23, 2015*, pages 3413–3422. ACM, 2015.

[15] S. A. Safdar, M. Z. Iqbal, and M. U. Khan. Empirical evaluation of uml modeling tools–a controlled experiment. In G. Taentzer and F. Bordeleau, editors, *Modelling Foundations and Applications*, pages 33–44, Cham, 2015. Springer International Publishing.

[16] T. I. Strasser, A. Zoitl, and G. Ebenhofer. 4diac - ein open source framework für verteilte industrielle automatisierungs- und steuerungssysteme. In K. Fähnrich and B. Franczyk, editors, *Informatik 2010: Service Science - Neue Perspektiven für die Informatik, Beiträge der 40. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Band 1, 27.09. - 1.10.2010, Leipzig, Deutschland*, volume 175 of *LNI*, pages 435–440. GI, 2010.

[17] M. A. Teruel, E. Navarro, V. López-Jaquero, F. M. Simarro, and P. González. A CSCW requirements engineering CASE tool: Development and usability evaluation. *Information & Software Technology*, 56(8):922–949, 2014.

[18] M. Vigo, C. Santoro, and F. Paternò. The usability of task modeling tools. In A. Z. Henley, P. Rogers, and A. Sarma, editors, *2017 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2017, Raleigh, NC, USA, October 11-14, 2017*, pages 95–99. IEEE Computer Society, 2017.