USABILITY EVALUATION OF SOFTWARE TOOLS FOR ENGINEERING DESIGN

Hashemi Farzaneh, Helena; Neuner, Lorenz

Technical University of Munich

ABSTRACT

Much of the work in design research focusses on the development of methods and tools to support engineering designers. Many of these tools are nowadays implemented in software. Due to the strongly growing use of computers and smart devices in the last two decades, the expectations of users increased dramatically. In particular users expect good usability, for example little effort for learning to apply the software. Therefore, the usability evaluation of design software tools is crucial. A software tool with bad usability will not be used in industrial practice. Recommendations for usability evaluation of software often stem from the field of Human Computer Interaction. The aim of this paper is to tailor these general approaches to the specific needs of engineering design. In addition, we propose a method to analyse the results of the evaluation and to derive suggestions for improving the design software tool. We apply the usability evaluation method on a use case - the KoMBi software tool for bio-inspired design. The case study provides additional insights with regards to problem, causes and improvement categories.

Keywords: User centred design, Computational design methods, Evaluation, Bio-inspired design / biomimetics

Contact:
Hashemi Farzaneh, Helena
Technical University of Munich
Laboratory of Product Development and Lightweight Design
Germany
helena.hashemi@tum.de

Cite this article: Hashemi Farzaneh, H., Neuner, L. (2019) 'Usability Evaluation of Software Tools for Engineering Design', in *Proceedings of the 22nd International Conference on Engineering Design (ICED19),* Delft, The Netherlands, 5-8 August 2019. DOI:10.1017/dsi.2019.136

1 INTRODUCTION

Much of the work in design research focusses on the development of methods and tools to support engineering designers. Many of these tools are nowadays implemented in software. Due to the strongly growing use of computers and smart devices in the last two decades, the expectations of users increased dramatically. In particular users expect good usability, for example little effort for learning to apply the software. Therefore, the usability evaluation of design software tools is crucial. A software tool with bad usability will not be used in industrial practice. Recommendations for usability evaluation of software often stem from the field of Human Computer Interaction. The aim of this paper is to tailor these general approaches to the specific needs of engineering design. In addition, we propose a method to analyse the results of the evaluation and to derive suggestions for improving the design software tool. In section 2, we therefore review approaches to usability evaluation from software development and from engineering design. We develop an engineering-design specific method in section 3 and apply this method on a use case - the *KoMBi* software tool for bio-inspired design in section 4. Section 5 concludes with learnings from the application of the method in the case study.

2 BACKGROUND: USABILITY EVALUATION

In this section, we firstly give an overview on common approaches to usability evaluation in general software development (2.1). To be able to adjust these approaches to the engineering design context, we secondly introduce approaches to usability evaluation of engineering design tools (2.2).

2.1 Software development

As the utilization of computer software and websites increased during the last decades, higher standards for its design processes were set. Regular and iterative evaluations help to realise changes in the early stages and hence save costs. In the context of Human Computer Interaction, the concept of "Usability" gained more and more importance. In addition, software has to fulfil functional requirements, similar to other technical systems. Nielsen (1993, p.24-25) refers to the fulfilment of functional requirements by the term "utility" whereas "usability" describes how well the functions can be used. As Figure 1 shows, both aspects contribute to the overall acceptance of software. When designing software, we have to bear in mind that an increased functionality can have a negative effect on usability as the software becomes more complex to use (Eason 1984, p.133-134)

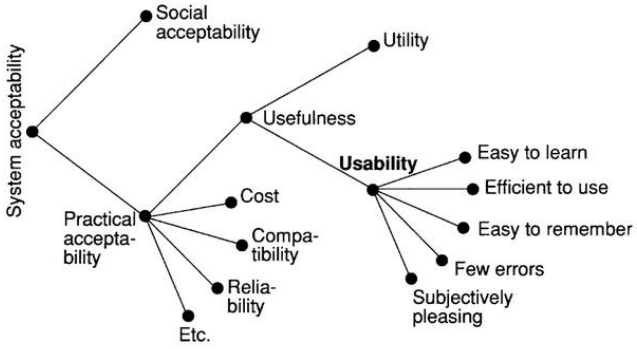

Figure 1: How usability determines the acceptance of a software system (Nielsen 1993, p. 25)

The standard ISO 9241-11 entitled "Ergonomic requirements for office work with visual display terminals" has become the main reference of usability in the domain of human-computer interaction (Jokela *et al.*, 2004, p. 155; Sarodnick *et al.*, 2006, p. 31). The standard defines usability as:
"The extent to which a product can be used by specified users to achieve specific goals with effectiveness, efficiency and satisfaction in a specified context of use" (ISO 9241-11, 1998).
A number of different approaches have been developed to evaluate the usability of software (Nielsen 1993, 223-226):
Formal evaluation methods refer to analytic, expert-based evaluation based on structured, formal models (e.g. GOMS) or guidance lists (e.g. EVADIS (Oppermann 1992)). The formal analysis of the

https://doi.org/10.1017/dsi.2019.136 Published online by Cambridge University Press

interface or tasks makes it possible to analyse usability aspects quantitatively. For instance, the GOMS method adds up the execution times necessary to fulfil a certain task such as "deleting a word". The time for mentally preparing for the task, moving the hand to the mouse, pointing to the word with the cursor, and pressing the delete button are added up to obtain the total execution time (Card, Moran and Newell 1983). This purely quantitative view supports the detection of some, but not of all usability problems.

Inspection methods (e.g. heuristic evaluation (Nielsen 1993, p.115-155), cognitive walkthrough (Lewis *et al.*,1990)) are a well-known analytic approach based on expert evaluators. Inspection methods are used at the early stages or the inner iterative steps of the design cycle. As inspection methods have a broader focus than formal evaluation methods, they can detect comparably many usability problems relying on only a few experts. A drawback is that no real users are involved. Thus, improvement potential for unexpected needs, particularly of unexperienced users, might not be revealed.

Observational techniques are used in form of follow-up studies or to analyse users' task assignments in a cost-efficient and simple manner. At least three test users are recommended. If the observer does not interfere with the user, almost real conditions of the set up are given. The discovered usability problems can result in new functions and features (Diaper 1989 p.210-237, Hix *et al.*, 1993, p.106, Nielsen 1993).

The *thinking aloud* technique (Lewis 1982) requires three to five users and is therefore a relatively cost-efficient method to detect users' misunderstandings and their reasons. However, the double load of speaking and working at the same time can disturb users. Additionally, verbalising issues might create false or unreal problems during the evaluation process.

Interviews are an indirect evaluation method which require approximately five users on average and can be conducted flexibly depending on the purpose of the study. In-depth interviews allow to discover real problems as well as feelings of a user. However, the subjective results of this time-consuming procedure are difficult to analyse, to code and to compare (Savin-Baden *et al.*, 2013, p. 357-367).

Questionnaires are likewise assigned to the category of indirect methods, as they collect subjective user data. However, they are more suitable for quantitative analysis if at least 20 to 30 users participate. The data collection can be replicated easily, once the questionnaire is established. The high level of standardisation and amount of validated surveys guarantee objectivity and reliability of a study. Nevertheless, the pilot study and validation of a questionnaire is resource- and time-consuming. For usability evaluation, a number of standard questionnaires have been developed, e.g. QUIS (Shneiderman 1987), SUMI (Kirakowski 1993) and the IsoNorm 9241/10 (Prümper 1999).

To summarise, the selection of methods for usability evaluation of software depends on the purpose of the evaluation (quantitative/ qualitative), the development stage and the availability of users. It is a common procedure to combine methods in order to balance their advantages and disadvantages (triangulation).

2.2 Engineering design

A frequently cited approach to designing research projects in engineering design research is the design research methodology developed by Blessing and Chakrabarti (2009). The methodology includes guidelines to evaluate so-called *supports*, i.e. all kinds of methods and tools. Blessing and Chakrabarti (2009, p. 37) divide support evaluation into two parts: The first part, the *application evaluation* addresses the usability and applicability of the support. The second part, the *success evaluation* focusses on the usefulness of the support. This categorization is in accordance with Nielsen's (1993) view on usability and usefulness (cf. Figure 1).

With regards to methods of data collection, Blessing and Chakrabarti (2009, pp. 254–273) point out two engineering-design specific methods in addition to common methods such as questionnaires, interviews etc.: *collecting documents* and *collecting products*. Both methods focus on the output of a (software) tool. In engineering design, tools must support the design of engineering products. Therefore for the evaluation of the tool, the quality of the designed artefacts, such as product models, prototypes or components of the final products are an important factor.

https://doi.org/10.1017/dsi.2019.136 Published online by Cambridge University Press

3 DEVELOPMENT OF THE USABILITY EVALUATION METHOD

From the literature on usability evaluation discussed in the previous chapter, we developed a method for usability evaluation of software tools for engineering design. Based on the general research questions on usability (3.1), we select data collection methods (3.2) and propose a data analysis procedure (3.3) tailored to engineering design.

3.1 Research questions for usability of software tools

As Figure 1 shows, the usefulness of a software tool is related to the two aspects utility and usability. Even though our focus is on usability, we still have to consider utility. Both factors are related - in order to perceive the utility of software, the user must be able to use it. On the other hand, if the software is easy to use, but has no utility, the users will not need it either. The research questions derived from these two major aspects of software evaluation can therefore be formulated as:
1. Utility: "How well supports the software tool the envisaged task and what are the major aspects to improve?"
2. Usability: "How easy to use is the software tool and what are the major aspects to improve?"
The research questions include a quantitative (*How well/ How easy…?*) and a qualitative aspect (*What are the major aspects to improve?*) (Gediga *et al.*, 2002, p. 127-153). If there are several software tools (or a software and a paper-based tool), the research questions can be reformulated to compare the different tools.

3.2 Data collection

In order to collect both quantitative and qualitative data for both research questions, we propose a mixed-method data collection. An additional aspect is to deliberately include the user's opinion on the software tool, but to also collect more objective data. This, in particular, enables a comparison between the data and makes it possible to filter out side effects (triangulation). The data collection methods described in section 2 are categorized in Table 1. For the evaluation of the usability of software tools, we considered three factors of particular importance:

Table 1: Overview on data collection methods (bold: selected methods)

Data collection methods	**Involvement of the user**	**Focus of analysis**	**Timing**
	No users involved/ analysis of process or products/ direct user feedback	*Quantitative or qualitative*	*During the usability test/ only retrospective*
Inspection methods	No users involved	Qualitative	During the usability test
Formal evaluation methods		Quantitative	
Observational techniques	Analysis of process or products	Qualitative	During the usability test
Collecting documents		Quantitative	
Collecting products			Only Retrospective
Thinking aloud	Direct user feedback	Qualitative	During the usability test
Interviews			Only Retrospective
Questionnaires		Quantitative	

The **degree of involvement of the user** ranges from none in the case of inspection methods and formal evaluation methods to direct user feedback. Methods that do not require users can be applied in the early stages of tool development when only experts are able to evaluate the prototype of the tool. If a software application has already been implemented, however, a combination of direct user feedback and the analysis of process and products are more useful. The comparison between subjective user feedback and an analysis of objective data, e.g. the time a user needs for a specified task, the use of specific elements of a software etc. is important to draw adequate conclusions.
The **focus of analysis** can be either quantitative or qualitative. Both are needed to evaluate the software and to identify improvement potential.

https://doi.org/10.1017/dsi.2019.136 Published online by Cambridge University Press

The timing of the data collection can be either during the usability test itself or in retrospective. Collecting data during the usability test is preferable. However, if direct user feedback is collected, this can negatively influence the user's concentration on the software tool.
For our usability evaluation of design software tools, we consequently select two methods for the analysis of process or products: *observation* (qualitative focus) and *collecting documents* (quantitative focus). Moreover, we select two methods for direct user feedback: *interviews* (qualitative focus) and *questionnaires* (quantitative focus). The interviews are to be conducted after the user fills out the questionnaire to avoid an influence of the interview on the quantitative ratings in the questionnaire.
Figure 2 shows the resulting procedure of data collection to evaluate the usability of a software tool for engineering design. Under the headline "action", the instruction of the user in a tutorial and the actual usability test by the user is depicted.

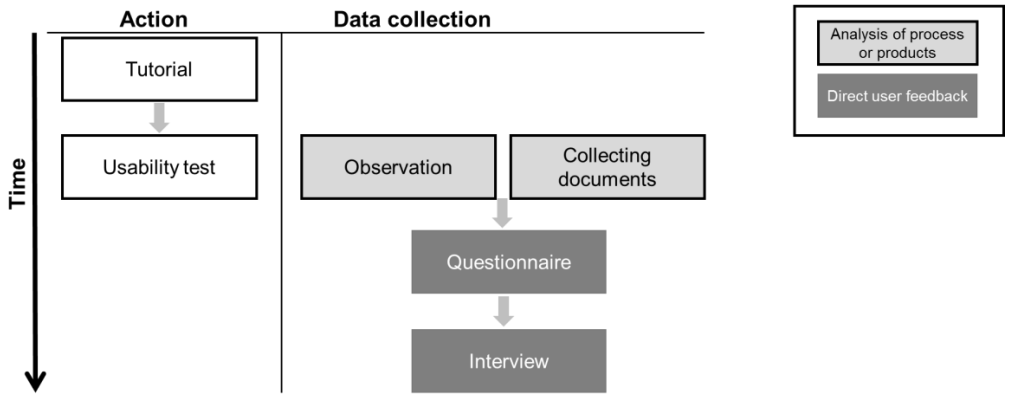

Figure 2: Data collection

3.3 Data analysis

We propose to start the analysis by examining the qualitative data from the observation and the interviews. The aim is to first analyse usability problems (part A) and then to check their criticality using the quantitative data (part B). The results of the usability evaluation can then be used for the suggestion of improvements (part C).
Part A, the analysis of usability problems, includes five steps:
Step 1: Identification of usability problems
By carefully regarding notes, transcriptions and videos of the observation, the researcher identifies reoccurring problems with the software tool.
Step 2: Categorization of usability problems
After having collected all usability problems, the problems must be categorized into broader categories to allow for a better overview on the problems.
Step 3: Identification of causes for the usability problems
For each usability problem, one or several causes are identified. If it is useful, the causes can also be categorized. Problems from the same category are likely to have similar causes.
Step 4: Ranking of usability problems based on severity and frequency
Nielsen (1993, p. 102-105) proposes to rate the criticality of usability problems based on severity and frequency.
The result of analysis of the usability problems is a qualitative understanding of the improvement potential of the software tool both regarding usability directly (research question 2) and regarding utility (research question 1).
Part B then rates utility and usefulness quantitatively. Here, the quantitative data from the collection of documents and the questionnaire is used.
1. Utility: Rating of the effectiveness and efficiency of the software tool with regards to the envisaged task
For this rating, the collected documents are analysed. To rate effectiveness, the researcher must develop quantitative measures for the fulfilment of the task. If possible, a comparison to a control group of designers who work without the software tool is recommendable. To rate efficiency, time or effort, applicable questions are: "did the users fulfil the task in the expected time frame?", or "did the users continuously create the content, they were expected to do?" etc.

https://doi.org/10.1017/dsi.2019.136 Published online by Cambridge University Press

2. Usability: Rating of the single "components" of the software
The questionnaires are used to rate single "components" of the software, such as subtasks, the design of the user interface, the interaction with the software etc. This analysis is conducted to verify the criticality rating of step 4 of the analysis of usability problems. In particular, it provides a comparison to the users' view on the software. If the users rate the criticality of problems very differently in comparison to the researcher, the reasons for this have to be explored.
Part C uses the results of the usability evaluation of the software tool for the suggestion of improvements:
Based on the identified causes (part A, step 3) and the criticality ranking (part A, step 4) counterchecked to the quantitative evaluation (part B), the researcher develops solutions in particular to the more severe usability problems. The effect of the changes has to be evaluated carefully, i.e. if the logic of the software is put at risk by the changes, the problem must be solved in a different manner.
In the following section, the usability evaluation is used on a concrete case.

4 APPLICATION ON THE *KOMBI* SOFTWARE

4.1 Use case: *KoMBi*

KoMBi - the **co**mmunication **m**odel for **bi**o-inspired design - is a modelling approach that aims at facilitating the communication between biologists and engineers. Regarding the level of detail and effort, *KoMBi* is situated in-between the *Biocards* (Lenau *et al.*, 2015) and the *SBF* (Goel *et al.*, 2009) or *SAPPhIRE* (Chakrabarti *et al.*, 2005) modelling approaches for bio-inspired design.
The core measure of the method is to provide graphical models which are easily understandable for biologists or engineers. Using the same modelling approach, both biologists and engineers are supposed to generate models which can be mapped to models from the other discipline. To create the graphical models, the two main steps of "system description" and "system behaviour & properties" have to be followed (Hashemi Farzaneh *et al.*, 2015)
A software tool implements both modelling steps. The usability of this software tool is evaluated using the evaluation approach developed in section 3.An exemplary model of one of the test users is shown in Figure 3.
The usability tests were carried out with ten participants to find the usability problems and major points of improvement regarding the two research questions. The participants were graduated biologists or engineers, most of them doctoral candidates and post-doctoral researchers from different mechanical engineering and biology laboratories of the Technical University of Munich. As *KoMBi* is designed to be used by engineers and biologists, five participants were selected from each group (purposive sampling). The amount of ten test users was chosen to assure a minimum probability of detecting present usability problems. According to Nielsen (1993) this probability amounts to 96% for ten and to 80% for five participants (assuming a detection rate of 28% for a single user).

4.2 Results of the usability evaluation of the *KoMBi* software

Figure 4 gives an overview on the usability evaluation of the *KoMBi* software.
In part A, 49 problems were identified (step 1) and categorized into five main and 12 subcategories (step 2).
The main categories were:

- Functional level: All problems directly related to the activities of the user. In the case of *KoMBi*, elements had to be generated and related by arrows. All problems related to the generation, connection or other activities are put into this category.
- Component level: All problems which can be assigned to a single "component" of the software, e.g. the elements, relations, the image (system behaviour) etc.
- Process: Problems which were related to the overall working process with the *KoMBi* software tool. An example is that some users had problems with using the elements they generated in the "system behaviour" step in the "system description" step.
- Interface: All problems concerning the visual design, e.g. the colouring of the elements, the size of the font, the size of the elements which could not be influenced by the users etc.
- Support: all problems that arouse from a lack of explanations for the users, e.g. missing explanation for certain "buttons".

https://doi.org/10.1017/dsi.2019.136 Published online by Cambridge University Press

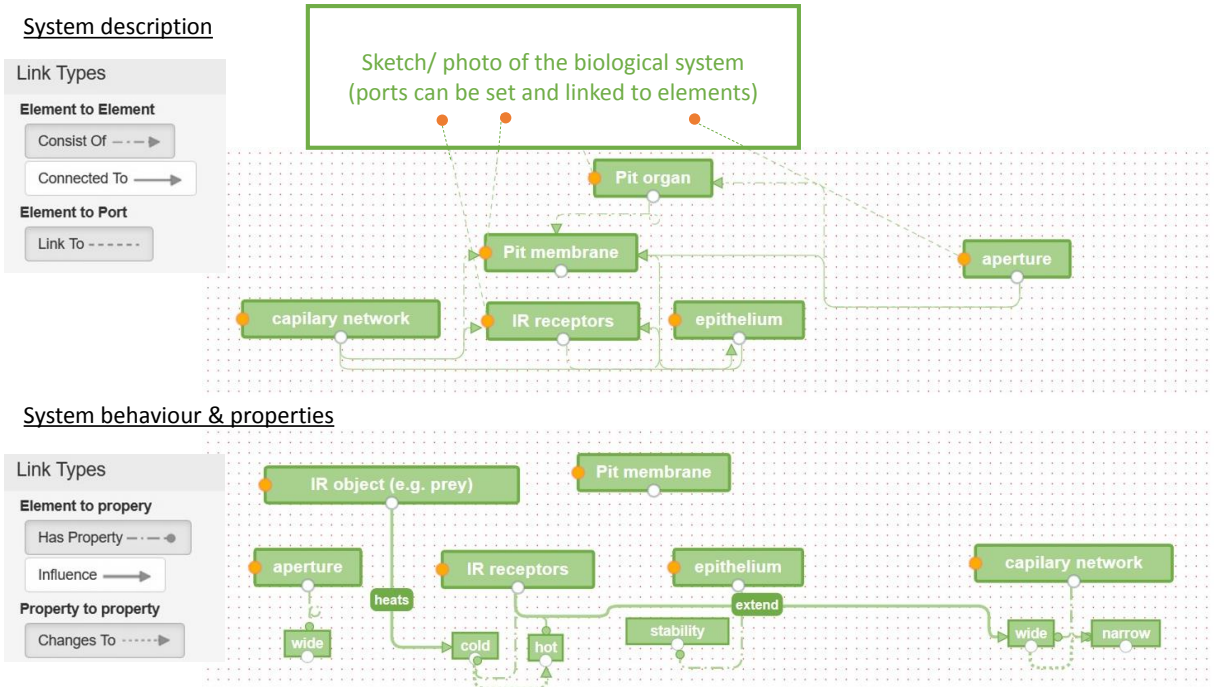

Figure 3: "Infrared sensing in snakes" - model a test user generated using the KoMBi software tool

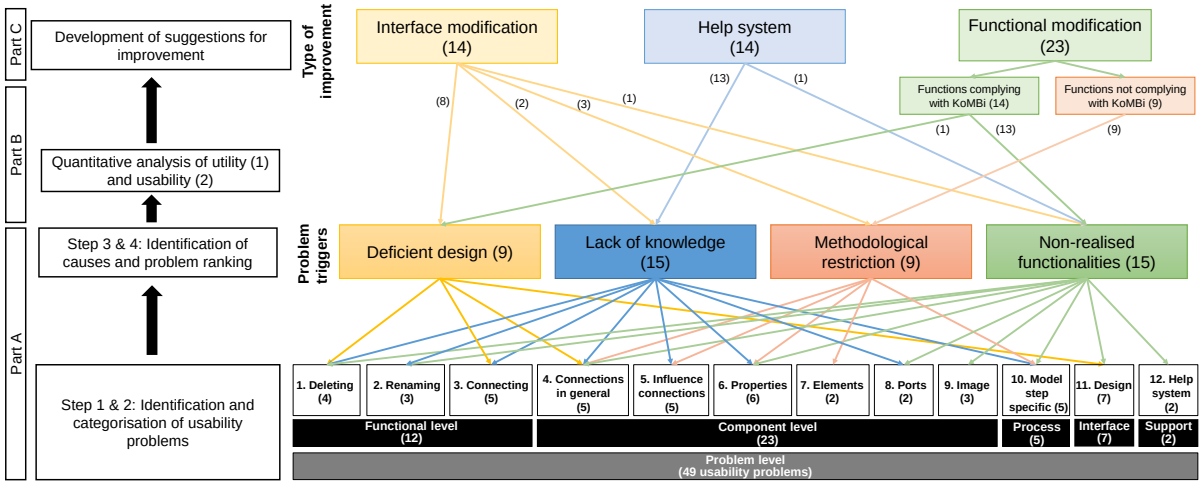

Figure 4: Results of the usability evaluation of the KoMBi software

In step 3, the problems were related to one of four causes. As Figure 4 shows, the main causes are:

- Deficient design (9 problems): Bad design of the user interface
- Lack of knowledge (15 problems): Too complicated user tasks or missing explanations
- Methodological restrictions (9 problems): restrictions of the underlying concept of *KoMBi*
- Non-realised functionalities (15 problems): Missing functionalities of the software

In step 4, the problems were ranked according to priority and frequency. As a result, the problems "incorrect relations between components" and "difficult selection procedure of relations" were ranked as the most critical.

In part B, the quantitative data was regarded.

As a quantitative measure for utility (research question 1), we analysed the *KoMBi* models generated by the users (data collection method: collecting documents). In particular, we examined the number of components (elements, relations, properties etc.) generated by the test users. Figure 5 shows the result. On average, for each element each user generated 7.07 components (directly connected relations, and properties). Hence, we predict that if a user needs five elements to model a technical or biological system, the average number of components will amount to 35.35 (five times 7,07).

https://doi.org/10.1017/dsi.2019.136 Published online by Cambridge University Press

In order to forecast the time needed, an average value of the user's efficiency has to be assumed. Regarding the ten test users, the mean value was 2.06 components per minute. Dividing the number of components by the estimated efficiency results in the forecasted time. Thus, 35.35 components are divided by 2.06 components/minute. This yields to a forecasted duration of 17.16 minutes for five created elements of a model. This forecast for the model duration is plotted in blue in Figure 5. The slope of the orange dotted linear regression line can be calculated by dividing 7.07 components/element by 2.06 components/minute. Thus, an average user needs 3.43 minutes per element. For instance, if a user generates six elements, the modelling time will be 20.6 minutes.
As Figure 5 shows, the majority of the orange crosses, indicating one test user each, are situated near the blue or the orange line. These users were able to generate elements with the software tool with approximately the average efficiency. A minority of three test users were outliers (one more efficient, two less efficient).

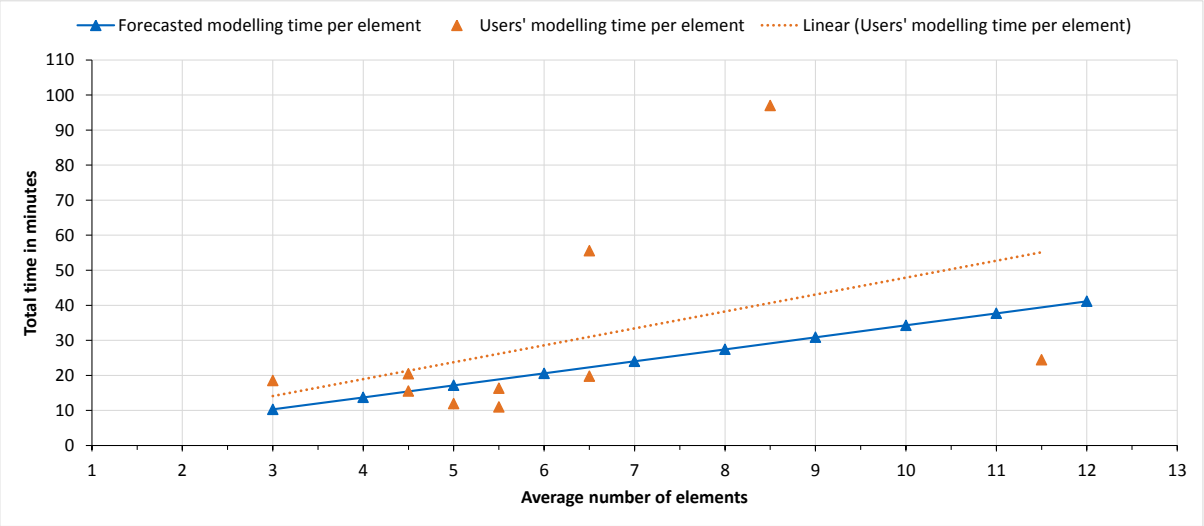

Figure 5: Measuring utility: efficiency of the software tool

As a quantitative measure for usability (research question 2), we analysed the questionnaires. The questionnaires based on ISO 9214/110 asked the users to rate the software tool on a scale between 1 (positive) and 5 (negative). As Figure 6 shows, the rating was overall positive (beneath 3) for all ISO 9241/110 categories. However, there were differences between biologists and engineers. One factor rated less positive than others by both engineers and biologists is *error tolerance*. If we compare this to the identified problems and their causes, this underlines the importance of removing in particular the problem cause *non-realised functionalities*. An example for this is that the tested software implementation of *KoMBi* does not allow to select elements or relations and to delete them. Elements or relations can only be deleted by clicking on a specific button.
In part C, suggestions for improvements are developed based on the problems. For *KoMBi*, the improvements can be grouped into the categories

- Interface modification: Changes of the user interface
- Help system: Additional support and explanations for the user
- Functional modification: Changes of the working process, the user has to follow. This category included suggestions that did not comply with the underlying logic of *KoMBi*. Therefore, they have to be evaluated carefully before implementing them.

Table 2 shows a list of suggestions to address the problems which had been rated most critical (part A, step 4) and which the quantitative evaluation had supported (part B). As can be seen in the table, the most critical utility problems are addressed by functional modifications. The most critical usability problems are addressed by interface modifications and a help system.

https://doi.org/10.1017/dsi.2019.136 Published online by Cambridge University Press

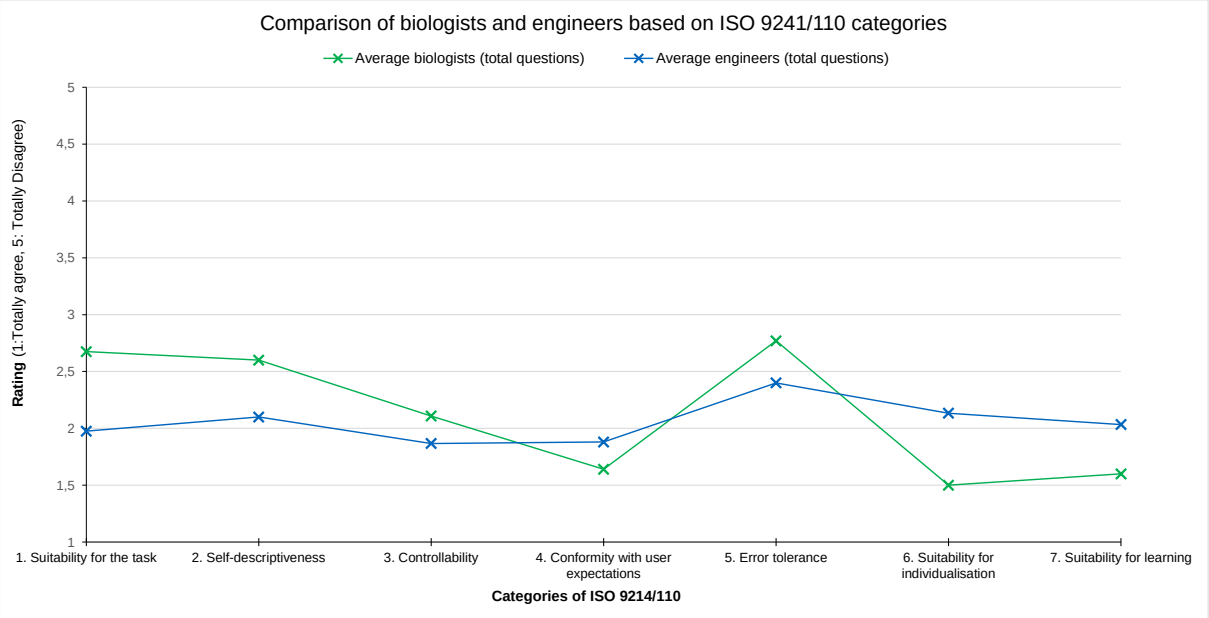

Figure 6: Rating of biologists and engineers according to ISO 9241/110 principles

Table 2: Suggestions for improvements of the KoMBi software tool to address the most severe problems

Research question	Suggested improvements to address the most severe problems
1) utility	**Functional modifications**: • Keyboard and mouse connection • Increased limit of text characters for renaming • Scaling of images
2) usability	**Interface modifications**: • Automatic line arrangement • Procedure of connecting components via relations **Help system**: • Software documentation • Concise explanatory pop-ups • Additional exemplary models

5 CONCLUSIONS

This work proposes a method for the usability testing of software tools for engineering design. The method was developed based on usability testing in software engineering and application evaluation in engineering design research. It proposes methods for qualitative evaluation (part A) and quantitative evaluation (part B). Moreover, based on a criticality ranking, improvements are suggested. The method was applied on a software tool for bio-inspired design (*KoMBi*). This provided additional insights on problem, cause and improvement categories:

- Problem categories: functions, components, interface, design, support
- Cause categories: deficient design, non-realised functionalities, methodological restriction, lack of knowledge
- Improvement categories: interface modifications, functional modifications, help systems

Interestingly, there is no one-to one mapping between problems, causes and improvements as one could expect (e.g. from problem category “design” to cause category “deficient design”). In future work, the method can be applied on further design software tools to evaluate if the problem, cause and improvement categories detected in our case study are useful in different contexts.

https://doi.org/10.1017/dsi.2019.136 Published online by Cambridge University Press

REFERENCES

Blessing, L.T. and Chakrabarti, A. (2009), *DRM, a Design Research Methodology*, Springer, London.

Chakrabarti, A., Sarkar, P., Leelavathamma, B. and Nataraju, B.S. (2005), "A functional representation for aiding biomimetic and artificial inspiration of new ideas", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)*, Vol. 19, pp. 113–132.

Card, S.K., Newell, A. and Moran, T.P. (1983), *The Psychology of Human-Computer Interaction*, CRC Press, New York.

Diaper, D. (1989), "Observation for Human-Computer Interaction", *Task Analysis for Human-Computer Interaction*, pp. 210–237.

Eason, K.D. (1984), "Towards the experimental study of usability", *Behaviour & Information Technology*, Vol. 3 No. 2, pp. 133–143.

Gediga, G., Hamborg, K. C. and Düntsch, I. (2002), "Evaluation of software systems", *Encyclopedia of Computer Science and Technology*, Vol. 45 No. 30.

Goel, A., Rugaber, S. and Vattam, S. (2009), "Structure, behavior, and function of complex systems. The structure, behavior, and function modeling language", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 23, pp. 23–35.

Hashemi Farzaneh, H., Helms, K. and Lindemann, U. (2015), "Visual representations as a bridge for engineers and biologists in bio-inspired design collaborations", In: Weber, C., Husung, S., Cantamessa, M., Cascini, G., Marjanovic, D. and Srinivasan, V. (Eds.), *Proceedings of the 20th International Conference on Engineering Design*, Milan, Milan, 27.–30.07.2015, Design Society, Glasgow, UK, pp. 215–224.

Hix, D. and Hartson, H. R. (1993), *Developing User Interfaces: Ensuring Usability Through Product & Process*, John Wiley & Sons, Inc., New York

ISO 9241-11. (1998), "Ergonomic requirements for office work with visual display terminals (VDTs) - Part 11 Guidance on usability".

Jokela, T., Iivari, N., Tornberg, V. and Electro, P. (2004), "Using the ISO 9241-11 definition of usability in requirements determination: case studies", In *Proceedings of HCI2004: Design for Life, the 18th British HCI Group Annual Conference*, Eds. Dearden, A. and Watts, L., Leeds Metropolitan University, UK.

Kirakowski, J. and Corbett, M. (1993), "SUMI: The software usability measurement inventory", *British Journal of Educational Technology*, Vol. 24 No. 3, pp. 210–212.

Lewis, C. (1982), "Using the "thinking-aloud" method in cognitive interface design", IBM Research Report RC9265, IBM Thomas J. Watson Research Center.

Lewis, C., Polson, P. G., Wharton, C. and Rieman, J. (1990), "Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces", In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.

Lenau, T., Keshwani, S., Chakrabarti, A. and Ahmed-Kristensen, S. (2015), "Biocards and level of abstraction", In: Weber, C., Husung, S., Cantamessa, M., Cascini, G., Marjanovic, D. and Srinivasan, V. (Eds.), *Proceedings of the 20th International Conference on Engineering Design*, Milan, Milan, 27.–30.07.2015, Design Society, Glasgow, UK, pp. 177–186.

Nielsen, J. (1993), *Usability Engineering*. Elsevier, Amsterdam.

Oppermann, R., Murchner, B., Reitner, H. and Koch, M. (1992), "Software-Ergonomische Evaluation", *Der Leitfaden EVADIS II*, de Gruyter, Berlin.

Prümper, J. (1999), "Test IT: ISONORM 9241/10", In: Bullinger, H.J. and Ziegler, J. (Eds.), *Human-Computer Interaction - Communication, Cooperation and Application Design* (pp. 1028–1032). Lawrence Erlbaum Associates, Mahwah, New Jersey.

Sarodnick, F. and Brau, H. (2006), *Methoden der Usability Evaluation*, Verlag Hans Huber.

Savin-Baden, M. and Major, C.H. (2013), *Qualitative Research: The Essential Guide to Theory and Practice*, Routledge, London.

Shneiderman, B. (1987), *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Addison-Wesley, Reading.

ACKNOWLEDGEMENTS

We want to thank all the participants from our user evaluation study, in particular Dr. Tobias Kohl from the Chair of Zoology at the Technical University of Munich who created the KoMBi model shown in Figure 3.

https://doi.org/10.1017/dsi.2019.136 Published online by Cambridge University Press