Usability Aware Model Driven Development of User Interfaces

Matthias Thiel[1], Andreas Petter[2], and Alexander Behring[3]

[1] NorCom Information Technology AG, Germany
matthias.thiel@norcom.de
[2] Telecooperation Group, TU Darmstadt, Germany
a_petter@tk.informatik.tu-darmstadt.de
[3] Telecooperation Group, TU Darmstadt, Germany
behring@tk.informatik.tu-darmstadt.de

Abstract. We propose an approach how to develop and integrate usability metrics in (multimodal) dialog systems and user interfaces with Model Driven Software Development (MDSD). It enables the developer to start early with simple metrics and allows him to refine them during the ongoing work. Additionally, we show how the resulting metrics can be used to estimate the usability in advance, so that the developer may care about critical parts of the application before testing it.

1 Introduction

The development of user interfaces by the aid of Model Driven Software Development (MDSD, e.g. [17]) technologies is a popular approach, especially when realizing web services. But in the field of multimodal user interfaces used in ambient intelligence, this approach is rarely used so far, resulting in few literature available on the topic. In this article, we want to get one step closer to automate usability evaluation by presenting an approach that allows integrating and developing usability metrics during the development process. The reason, why usability metrics should be considered to be an important aspect of the development process has been explained a lot of times [12]. Currently, the only reliable way to verify the usability is to perform user studies for the particular application. The disadvantages are high costs and that evaluation can only take place after development of the application has been almost completed [18,4]. One common approach to evaluate the usability of interactive systems is the analysis of log files for getting statistical values (e.g. timings) that are used for measuring usability or guessing the major usability problems. E.g. the approach presented in [15] derives Petri nets from log files that serve as foundations for applying usability metrics.

By introducing usability awareness into the development process analysis can be partially done automatically during development. Thus, the developer can watch usability metrics during development and may react on them as soon as problems are reported.

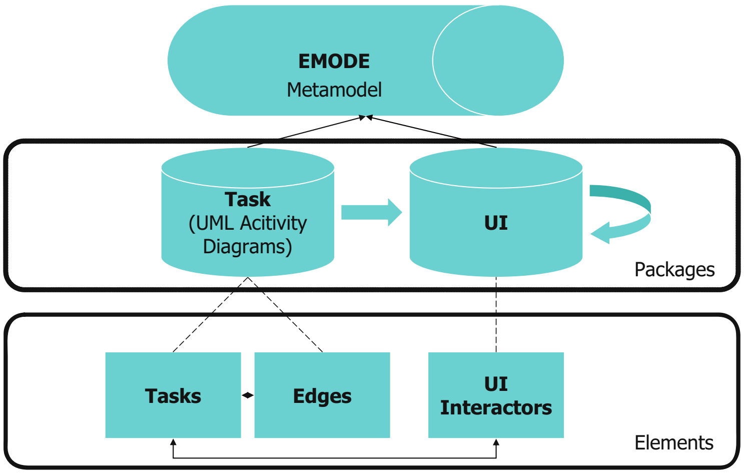


Fig. 1. EMODE Metamodel Overview

In section 2, a short overview of the meta models used is given. Models are introduced and the type of model editors which should get usability aware. In section 3, a new approach is presented for estimating usability in advance and refined in 4. Section 5 presents a small example for applying the new approach.

2 Meta Models

This section gives a brief overview about the meta models of the EMODE project [6] which we will use as an example for applying our usability metrics. An overview of the most important metamodel packages is given in figure 1. It is to be mentioned that the following definitions are not absolutely matching the EMODE meta model definition. They should just illustrate the concepts.

The meta models that are of relevance here are the task model and aui model. The main elements of the task model are tasks which can be subdivided into different types. The interesting type is the interaction task which will be associated to elements of the aui model, especially the interactor elements. An interactor element may be seen as a subtask that describes a needed information exchange between the user and the system, also called interaction step. Accomplishing the interaction steps will fulfil the associated interaction task.

3 Usability Evaluation

In this section an approach for estimating usability of interactive systems is proposed. Beginning from an abstract level metrics will be derived that are suitable to be applied on model transformations.

According to [9] a lot of the analysis methods presented in literature are to be used manually after software development has been finished. In contrast, the approach presented here tries to predict usability by investigating models before they are transformed into code.

Definition 1. *Usability analysis can either be **post-development** or **predictive**. Post-development analysis is applied after development, while predictive analysis is applied during development.*

When regarding usability, it is assumed that the user wants to reach one or more goals during the interaction with the system. Additionally, it is assumed that the goal is well-defined and reachable within the application. For getting to the desired dialog state the user has to invest some effort by performing one or more actions.

Definition 2. *Usability is inversely proportional to the user's effort in reaching his goal.*

Similar to the ideas in [5] and [14] we define:

Definition 3. *Concerning the time dimension (which is used often as a major metrics in post-development usability analysis) the user's effort for one interaction step can be structured in following phases:*

P_1 *Perception of the system output*
P_2 *Understanding the output and integrating it into the user's context*
P_3 *Inferential thinking processes for determining the next reaction*
P_4 *Uttering the next reaction*

Calculation of these efforts depends on the user and therefore it is not possible to perform them without a user model.

Definition 4. *A user model can be given **explicitly** by a formal model in machine readable format or **implicitly** by providing it within the algorithm to compute the metric.*

Potential problems of implicit user models are that they may not be argued upon, as they are not precise and consistent, and that they must be very well understood by developers and designers. However, most metrics are given without explicit user models. The reason why one would not define an explicit user model is that there is no agreement of how a meta model for a user model should look like.

Metrics are mostly based on evaluations of statistical values about usability properties derived from them. Therefore, evaluators cannot provide a user model without significant extra research and much effort would have to be invested to additionally create a user model for the metrics. Instead of proposing an elaborated user model here, it is proposed to figure out the user model iteratively by looking at the metrics involved and the needs of the applications to be developed.

3.1 Example

In the following a simple *example* is presented how to use the four phases involved and the development or integration of new metrics into the calculation of the overall usability evaluation value.

Assume the system output is: *"What is the color that the clocks should have and how many do you need?"* As noted above, the developer wants to estimate the effort that the user has to perform for reacting to this system output. For now, he will use an implicit user model for getting faster to a first draft of his metrics, which could look like this:

- P_1 It is assumed that the user's effort somehow depends on the amount of words he has to perceive. Therefore we define the effort for P_1 to be $E(P_1) := m$ *(amount of words).*
- P_2 Similar to P_1 we count the concepts that the user has to map to his domain model. In this case, the concepts *color* and *clock* are candidates. For now, we ignore relationships between them.
- P_3 Like most of the human computer interfaces, it is assumed that the user already knows what he wants. The system just asks for known facts, like an html form does. This means $E(P_3) := 0$.
- P_4 The system could estimate the amount of words the user might need for answering. For example, the answer of the user could be: *"I need three ones in blue."*. The general formula might be $E(P_4) := m$ (*average amount of words that the user utters*). Taking the example as average case, we get $E(P_4) = 6$. For getting a better estimation of the average case this could be retrieved (automatically) by analyzing the corresponding (speech) grammar.

After having determined the efforts for each of the four phases, the developer wants to summarize the particular efforts. However, at the current stage of development of the proposed approach this turns out to be very difficult, since metrics are not normalized and the developer needs to normalize them manually. This means that the effort for processing at P_1 is not comparable with the effort of mapping a concept to the domain model at P_2 without further research by the developer. A simple approach is to sum the weighted efforts per phase. This reduces the problem to how the weights w_i are to be estimated for calculating the total effort for each interaction step

$$\sum_{i=1}^{4} w_i \cdot E(P_i) \tag{1}$$

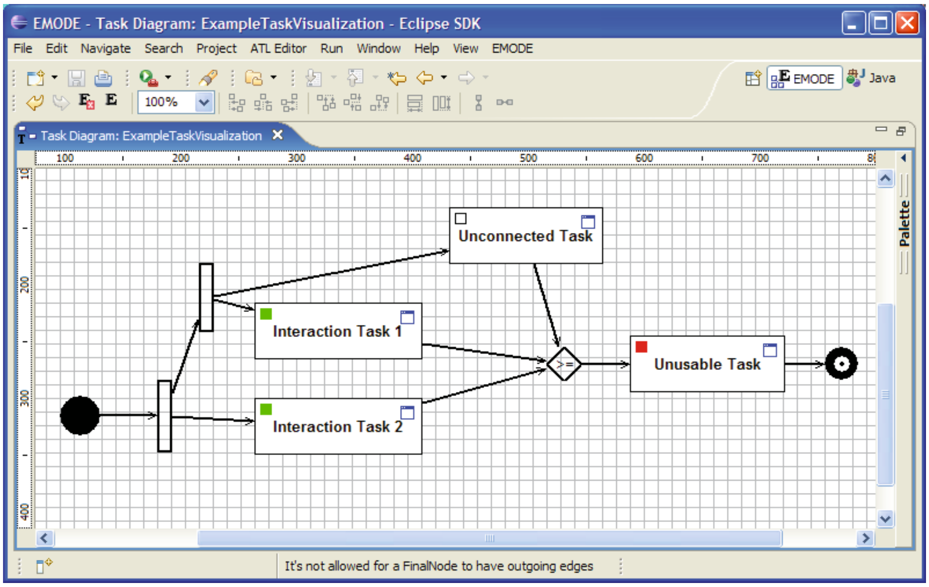
The determination of the weights strongly depends on the definition of the functions, because the range of values could be very different. The user model plays an important role here, if the user will have to face difficult circumstances (e.g. a noisy environment or a disability) in a phase.

Most of dialog manager components integrate the four phases into a single transition from one dialog state to the next one [3]. This may be seen as the basis on which most of the definitions of existing usability metrics are based on.

3.2 Implementation of the Framework

The current state of the art is to use style guide analyzers [7] to tell the developer which parts of the user interface have to be improved. To provide better support for usability to developers, the implementation presented here applies metrics and presents hints within the task editor for maximum visibility. The framework which is needed to integrate metrics into the development environment has already been implemented. The implementation uses the EMODE models presented in section 2, but may be applied to any other meta model, which includes a task model and a user interface model (such as [10,16]). The idea is to employ usability metrics in measures used in rules for model-to-model transformation (this is similar to, or an implementation of, the definition "usability preserved by transformation" given in [2]) or in the editor to be an aid for the developer. Currently the latter has been implemented (see figure 2).

First, each of the metrics defined are calculated and summed up as presented in the previous section. Then, each task which may be connected to a user interface component is annotated by a small rectangle with changing colours. Green rectangles symbolize that the user interfaces connected to the tasks are usable according to the implemented metrics, gradually ranging to red rectangles showing that the associated user interfaces are not as usable as the ones marked using green. As the task model is central to the development process, a centralized view is provided to the developer, who then can edit the user interfaces which are considered to be not as good (more red rectangles) as the others (more green rectangles). In lack of a standard language to describe usability


EMODE - Task Diagram: ExampleTaskVisualization - Eclipse SDK
File Edit Navigate Search Project ATL Editor Run Window Help View EMODE
100%
EMODE Java
Task Diagram: ExampleTaskVisualization
Palette
Unconnected Task
Interaction Task 1
Interaction Task 2
Unusable Task
>=

It's not allowed for a FinalNode to have outgoing edges

Fig. 2. Development Environment

metrics in specialized declarative or imperative languages in a flexible way, the implementation of the metrics is based on JAVA.

4 Usability Metrics During Application Development

There exist several options, how to apply usability metrics during application development. So far four types have been identified, which may be seen as to build on top of insights gained from usability studies as shown in figure 3. Each bar of the pyramid corresponds to a period during the development process.

The first approach, at the bottom of the figure, involving at least one **user study**, has been applied practically very often. After having designed a mock-up or a prototype user interface, a usability evaluation is done by asking questionnaires or logging user actions while performing tasks. Then, the whole process re-iterates as the developers improve the user interfaces using the evaluation outcome. As a second approach, a **post-development** metric (level 2 in the figure) may be derived from the knowledge gained from the user studies. Applying a post-development metric allows for usability evaluation without asking real users. This may not always be possible, as knowledge may not always be generalized from user studies in a way that it may be formalized, or the knowledge may apply to specific situations only. However, it is always possible to evaluate the usability against real users.

The third approach, which is propagated here, involves the definition of metrics on application models (**predictive metrics**, level 3). These metrics are either drawn from literature or created by analyzing previously performed user-studies, which present metrics for applications. These metrics will reduce turnaround times as tests are performed on models, such that the application does not need to be developed, but its model only.

These metrics may then be refined and improved to be suitable for the fourth approach (**constructive metrics**) by thinking about properties, which directly influence the value of the metric, such that they may be used to directly develop the models with optimum performance concerning the metric. These constructive plans prevent developers from developing models, which are not usable according to the metric. While being the most efficient approach for developing models, it is not always possible to derive model elements or attributes which should be changed to optimize the value of a metric, e.g. the predictive metric is only defined using a heuristic measure. As these metrics are algorithms, they may even be applied to model transformations, e.g. [8].

A lot of works in the research area of usability metrics have been published on the topic of user studies (e.g. [9,13]). Some works try to assess usability after development by applying post-development metrics (e.g. [15]). Other works try to predict usability by evaluating guidelines [7], which may be seen as a predictive metric. [8] presented a set of rules for transformations which has been designed to support several constructive metrics.

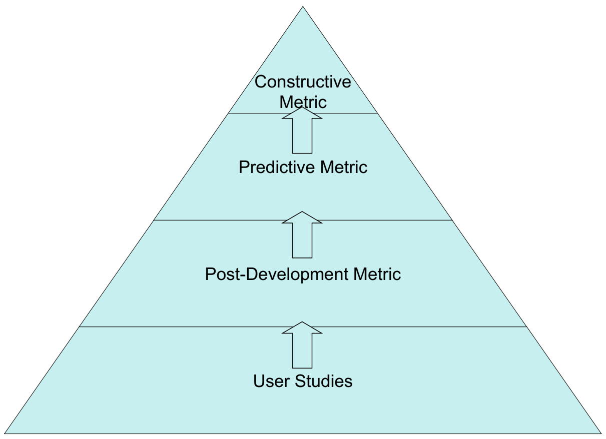


Fig. 3. Types of Metrics

In fact, usability metrics are a formalization of usability quality factors and criteria as optimization problem. We currently try to refine the factors defined in [1,19]. [1] defines quality factors like multimodality and recoverability, where multimodality means to give the user the opportunity to choose the appropriate modality at any time for providing his input. Metrics may be derived, e.g. estimating usage of the modality dependent interaction phases can serve as a foundation for predicting, what kind of set of modalities will be more usable for a given dialog situation. [19] enumerates unambiguousness as a usability attribute. An established way of ensuring this is to insert confirmation questions into the dialog. On the other hand, using too many confirmation questions is in conflict to an efficient dialog (in sense of a fast walkthrough). By estimating the user effort, it is possible to reason about the frequency of confirming facts.

5 Example Metric

In this section, it is shown how to integrate known metrics [11] into the $E(P_1)$ definition of section 3.1. The calculation is done for a particular interaction phase and uses specific knowledge about interactors or tasks. Starting from the example $E(P_1) := m$ given above, further metrics will be added step by step. m measures efforts for voice modality while the new metrics aspect ration (α) will measure efforts for visual modality.

α is the ratio of the height of a dialog box to its width. Desireable values are between 0.5 to 0.8 (ca. 0.65). Therefore, the extended formula is

$$E(P_1) := g_1(m) + g_2(\|\alpha - 0.65\|) \tag{2}$$

where g_i are functions for normalizing and weighting the particular efforts.

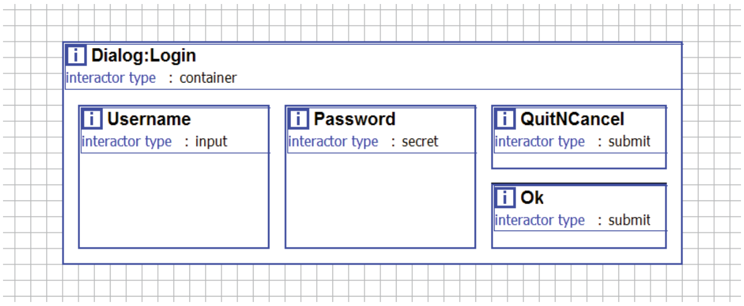


Fig. 4. Example of AUI model for login dialog

In 4, an example of the aui model diagram for a login dialog is shown. The resulting login dialog uses voice and graphical modalities. The voice modality asks for username and password with the phrase: "What is your name and password, please? ", while the graphical modality shows a usual login dialog box (in our case with the size of 300x150 pixels). The developer wants to calculate

$$E(P_1) = g_1(m)+g_2(\|\alpha-0.65\|) = g_1(7)+g_2(\|\frac{150}{300}-0.65\|) = g_1(7)+g_2(0.15) \quad (3)$$

In this case defining the normalization functions is straightforward. $g_i(x)$ may be defined by $g_i(x) = x$. The dialogue will suffer from large amounts of words being output, but is flexible towards α. However, bounds do exist. To produce high pain factors for extreme dialog box sizes (imagine a dialog box of 1000x1 pixels in size) g_2 is extended by: $x < 0.5 \vee x > 0.8 => g_2(x) = 1000$. Since this is not the case, the result is $E(P_1) = 7 + 0.15 = 7.15$.

6 Conclusion

An interaction step has been defined to be composed of four interaction phases. Using these phases an example was provided which shows how to calculate usability for interaction steps by giving metrics for each phase.

Distinguishing post-development, predictive and constructive metrics helps to select metrics for the evaluation process. The approach presented displays the outcome of predictive metrics in the task model to give the developer an overview of the usability of his application.

We feel that the approach presented seems promising to optimize the costs of developing usable interfaces. This is achieved by defining metrics and informing the developer about the usability of his user interfaces on an abstract level.

In the future we want to get more experience in defining usability metrics and especially the weighting of the different phases. The comparison of usability for different modalities will also be an issue. Additionally, model transformations should use predictive metrics for optimizing their resulting models.

References

1. Abowd, G.D., Coutaz, J., Nigay, L.: Structuring the space of interactive system properties. In: Proceedings of the IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction, pp. 113–129. North-Holland, Amsterdam (1992)
2. Abrahão, S., Iborra, E., Vanderdonckt, J.: Usability evaluation of user interfaces generated with a model-driven architecture tool. In: Law, E., Hvannberg, E., Cockton, G. (eds.) Maturing Usability: Quality in Software, Interaction and Value. HCI Series, vol. 10, pp. 3–33. Springer, Heidelberg (2007)
3. Balme, L., Demeure, A., Barralon, N., Coutaz, J., Calvary, G.: CAMELEON-RT: A Software Architecture Reference Model for Distributed, Migratable, and Plastic User Interfaces. In: Markopoulos, P., Eggen, B., Aarts, E., Crowley, J.L. (eds.) EUSAI 2004. LNCS, vol. 3295, pp. 291–302. Springer, Heidelberg (2004)
4. Beringer, N., Kartal, U., Louka, K., Schiel, F., Türk, U.: Promise - a procedure for multimodal interactive system evaluation. Technical Report 23, Ludwig-Maximilians-Universität München (May 2002)
5. Card, S., Moran, T.P., Newell, A.: The Psychology of Human-Computer Interaction. Lawrence Erbaum Associates, London (1983)
6. Dargie, W., Strunk, A., Winkler, M., Mrohs, B., Thakar, S., Enkelmann, W.: Emode - a model-based approach to develop adaptive multimodal interactive systems. In: Proceedings of the Second International Conference on Software and Data Technologies, Barcelona, Spain (July 2007), ISBN: 978-989-8111-09-8
7. Farenc, C.: ERGOVAL: une méthode de structuration des règles ergonomiques permettant l'évaluation automatique d'interfaces graphiques. PhD thesis, Université Toulouse (January 1997)
8. Florins, M.: Graceful Degradation: a Method for Designing Multiplatform Graphical User Interfaces. PhD thesis, Université catholique de Louvain (2006)
9. Ivory, M.Y., Hearst, M.A.: The state of the art in automating usability evaluation of user interfaces. ACM Comput. Surv. 33(4), 470–516 (2001)
10. Limbourg, Q.: Multi-Path Development of Multimodal Applications. PhD thesis, Université Catolique de Louvin (2004)
11. Mahajan, R., Shneiderman, B.: Visual & textual consistency checking tools for graphical user interfaces. (CS-TR-3639) (1996)
12. Nielsen, J.: Guerrilla HCI: using discount usability engineering to penetrate the intimidation barrier, pp. 245–272. Academic Press, Inc., Orlando, FL, USA (1994)
13. Nielsen, J.: How to conduct a heuristic evaluation. Internet (December 1994), `http://www.useit.com/papers/heuristic/heuristic_evaluation.html`
14. Norman, D.: The Design of Everyday Things. Currency (February 1990), ISBN 0-38526-774-6
15. Rauterberg, M.: A method of a quantitative measurement of cognitive complexity. In: Bagnara, S., van der Veer, G.C., Tauber, M.J., Antalovits, A. (eds.) Human-Computer Interaction: Tasks and Organisation, Roma, pp. 295–307. CUD (1992)
16. Sottet, J.-S., Calvary, G., Favre, J.-M.: Models at run-time for sustaining user interface plasticity. In: Proceedings of the Workshop Models at Run Time, Glasgow (October 2006)

17. Stahl, T., Völter, M.: Modellgetriebene Softwareentwicklung, 2nd edn., Dpunkt Verlag (May 2007), ISBN 3-89864-448-0
18. Walker, M.A., Litman, D.J., Kamm, C.A., Abella, A.: PARADISE: A framework for evaluating spoken dialogue agents. In: Cohen, P.R., Wahlster, W. (eds.) Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics, Somerset, New Jersey, pp. 271–280. Association for Computational Linguistics (1997)
19. Winter, S., Wagner, S., Deissenboeck, F.: A comprehensive model of usability. In: Proc. Engineering Interactive Systems 2007, Salamanca, Spain, Springer, Heidelberg (2007)