Evaluation of user interface adaptation strategies in the process of model-driven user interface development

Kai Breiner, Volkmar Gauckler
University of Kaiserslautern
Software Engineering Research Group
Gottlieb-Daimler Str.
67663, Kaiserslautern, Germany
breiner@cs.uni-kl.de,
volkmar.gauckler@gmx.net

Marc Seissler
University of Kaiserslautern
Center for Human-Machine-Interaction
Gottlieb-Daimler Str.
67663, Kaiserslautern, Germany
Marc.Seissler@mv.uni-kl.de

Gerrit Meixner
German Research Center for Artificial Intelligence (DFKI)
Trippstadter Str. 122
67663, Kaiserslautern, Germany
Gerrit.Meixner@dfki.de

ABSTRACT
In this paper, we describe the evaluation of our prototype Universal Control Device (UCD), which enables the control of various devices in modern dynamic production environments, while being able to adapt itself to the current configuration of the environment. While it is hard to apply traditional user interface design heuristics in recent paradigms – such as Ambient Intelligence – there is a demand for suitable compensation strategies addressing usage errors, which can be met by applying an adequate adaptation strategy. In a pilot study, we gained experience regarding differences in the performance of selected adaptation strategies in the case of our demonstrator.

Author Keywords
Universal Control Device, Adaptation Strategies, SmartFactory.

ACM Classification Keywords
H5.2. Information interfaces and presentation (e.g., HCI): Evaluation/methodologie, Prototyping, User-centered design.

INTRODUCTION
The ongoing technological development of microelectronics and communication technology is leading to more pervasive communication between single devices or entire pervasive networks of intelligent devices (smart phone, PDA, netbook, etc.). Especially industrial devices and applications can take advantage of modern smart technologies, e.g., based on ad-hoc networks, dynamic system collaboration, and context-adaptive human-machine interaction systems. The vision of Mark Weiser [1] concerning ubiquitous computing – also in production environments – is becoming a reality [10].

In today's production environments, technical devices often stem from multiple vendors using heterogeneous user interfaces that differ in terms of complexity, look&feel, and interaction styles. Such highly complex and networked technical devices or systems can provide any information at any time and in any place. This advantage can turn out to be a disadvantage when information is not presented properly according to the users' needs. This leads to problems, especially concerning the usability of the user interface. The level of acceptance of a user interface largely depends on its ease and convenience of use. A user can work with a technical device more efficiently if the user interface is tailored to the users' needs, on the one hand, and to their abilities on the other hand. Therefore, providing information in a context- and location-sensitive manner (depending on user, situation, machine, environmental conditions, etc.) has to be ensured.

Hence, in the following we will give a short introduction to the SmartFactory^{KL}, which serves as a demonstration environment for future intelligent production environments, and a Universal Control Device (UCD), which provides holistic control to various devices in such environments. Further, we give a brief introduction to user interface adaptation, usage errors, as well as to their compensation. After presenting our idea of how to approach compensation by using adaptation strategies, we describe the set-up of the corresponding controlled experiment and the preliminary results of the pilot study conducted. We conclude with the interpretation of how the results contribute towards our hypothesis.

SmartFactory^{KL}
Besides these aspects, modern production environments are characterized by a modular layout. Entire modules can be replaced or reorganized. Furthermore, these environments are able to react to errors occurring in the production process (e.g., device malfunction) and to dynamically reorganize parts of the process in order to ensure the production process. Thus, this also affects the user who

Pre-proceedings of the 5th International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2010): Bridging between User Experience and UI Engineering, organized at the 28th ACM Conference on Human Factors in Computing Systems (CHI 2010), Atlanta, Georgia, USA, April 10, 2010.

interacts with the individual devices – the user's workflow will change, or devices will not be available anymore.

Serving as a demonstration environment, the SmartFactory^{KL} in Kaiserslautern, Germany is able to simulate such a process. In previous work, we developed a UCD, which is able to provide access to various devices of the SmartFactory^{KL} [3][4].

Universal Control Device (UCD)

As a result of a model-driven process, the user interface of the UCD is being generated at run-time [2]. Starting with a topological description of the environment, enriched with user tasks on the single devices and information about how to address the single devices, this information is sufficient for generating a functional user interface [2][4]. In order to remain functional to the user, this user interface has to correspond to the current configuration of the environment and is additionally restricted to the functionality as specified in the underlying model. During field studies, faced by the need to adapt the user interface, we encountered the demand for a systematic strategy that would support the user as much as possible [3]. This was the trigger for a study on adaptation, which we will describe in the following.

ADAPTATION

After giving a brief overview of different types of adaptation – their properties as well as their impact on the users' workflow – we will elaborate types of usage errors resulting from static user interfaces and how adaptive user interfaces contribute to the compensation of such errors.

Static versus Dynamic User Interfaces

On the one hand, one important usability quality attribute is *memorizability*. The ease to remember helps users to speed up the process of interacting with the user interface [5][6]. Since humans have a very visual memory, the way a certain user interface is structured is essential for finding items faster. After a while, users form a coherent model of the user interface and are able to recall how to execute their workflow. If the system (and therefore the user interface) changes frequently, the user will not be able to form such a model [7][8]. On the other hand, there are user interface heuristics demanding that the user interface matches the real world [6]. In order to provide control over devices in dynamic environments – such as the SmartFactory^{KL} – it is extremely vital to match the user interface to the real world in order to remain functional.

These simple rules about how to develop a user interface appear to be contradictory for future information systems in our application domain. Hence, there is a need for an adaptation strategy that does not violate usability quality attributes leading to usage errors.

Usage Errors

In general, for each kind of usage error, there is a basic cause (see Figure 1). Using the right compensation technique – such as adaptation of the user interface – the users can be actively supported in preventing usage errors. Basically, there are two kinds of operating errors that may lead to a failure of the system.

In the first case, there are *slips*. Here, the user had the correct intention but executed the task in the wrong way. Most often this is caused by poor physical skills, or by the user interface just being just inadequate for use (e.g., buttons too small). In the second case – which is more interesting in our example – there are *errors*. Errors are characterized by the user having the wrong intention while executing the task. This is caused by a misunderstanding of the user interface (e.g., if the user interface is offering control over devices that are not available physically).

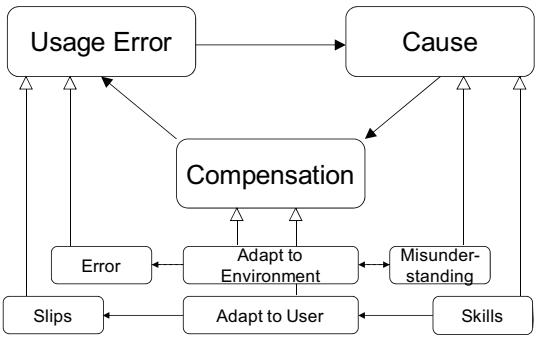

Figure 1. Relationship between usage errors, their causes, and compensation.

Compensation

Depending on the type of usage error, there are different ways to compensate in order to minimize the effect. *Slips*, which are caused by poor skills of the user, can be prevented by adapting the user interface to the user. In case of our application domain – intelligent production environments – we are dealing with predefined user roles and user groups and are therefore able to tailor the user interface to the needs of these user groups.

In dynamic environments, *errors* can be prevented by adhering to design heuristics as mentioned earlier. The system has to represent the current configuration of the user interface, while supporting the development of a mental model. Hence, a method of adaptation needs to be chosen that contributes to these heuristics.

Types of Adaptation

There are different methods of how an adaptation of the user interface can be executed. These methods differ in their way of execution as well as in their degree of intrusion into the users' workflow. In case of the UCD, a simple adaptation scenario would be the appearance or disappearance of devices in the device selection list. In the following, we will refer to this example while giving details about the individual methods.

The first method – which was implemented initially and led to the investigation described in this paper – is *ad-hoc* adaptation. Here, devices are added or removed from a device list according to the physical status of the respective

devices. Unless a regular user permanently observes the device list, he or she will not notice any change. Furthermore, this will be distracting for the user, because the structure of the user interface changes without notification.

Providing more information about the system state leads to the second adaptation method – *notification*. Now, the user interface provides information about the change and therefore supports the user in adapting his or her mental model. In case of the device list example, we implemented this method by applying the so-called instant-messenger metaphor [9]. This means that new or defective devices will be emphasized graphically, which provides information for the user to understand the current state of the system. However, the user may just overlook the notifications if he or she is distracted, and then the system cannot provide further support.

Thus, we implemented a third adaptation strategy, which aims at *confirmation* by the user. Here, the user has to actively confirm the change to the user interface. Referring to the device list of the example above, this strategy was implemented in terms of a dialog box asking for confirmation from the user that he or she has actually noticed the adaptation of the system. Thus, the system can be almost sure that the user is aware of the adaptation and is supported in the presumably most adequate way. A negative side-effect of this strategy is that the confirmation dialog may distract the user during his/her regular workflow.

Compensation

Each of the adaptation strategies differs as to how much it contributes to the compensation of possible usage errors.

Ad-hoc adaptation ensures consistency between the controllable environment and the corresponding user interface in order to prevent usage errors with respect to outdated user interfaces. But this approach provides no active support to the user at all.

Besides consistency with the current configuration of the environment, notification provides limited feedback to the user (e.g., by visually emphasizing new devices). Due to the fact that this communication with the user is unidirectional, such notifications can be easily overlooked by the user.

Confirmation provides all the functionality of the first two approaches on the one hand and demands confirmation by the user on the other hand. Thus, the user interface is aware of the fact that the user has recognized the new configuration and can therefore proceed with the regular functionality.

Hypothesis

According to the diverse properties of these strategies, we wanted to investigate which is the most adequate one in the case of our model-driven approach and therefore we formulated hypotheses that we are going to verify or falsify initially in a preliminary study described below. The hypotheses are tailored to the specific set-up of the controlled experiment.

H1. Effectiveness – completion rate
We assume that, on average, at least 85,6% of the test tasks can be completed without help.

Explanation: This hypothesis assumes that test persons can deal with the user interfaces and have understood their tasks and therefore can complete at least 6 out of the 7 tasks.

H2. Effectiveness – assistance
100% of the given tasks can be completed (with help – if needed).

Explanation: The user interface ensures consistency between environment and visualization independent of the adaptation strategy. Therefore, it should be possible to complete each task.

H3. Efficiency – strategy performance
Confirmation outperforms notification, which outperforms ad-hoc adaptation.

Explanation: On the basis of the different attributes elaborated earlier, we assume this ranking according to the performance of the different strategies.

EVALUATION – PILOT STUDY

To evaluate these hypotheses, we decided to conduct a controlled experiment. We implemented three instances of our model-driven process [2][3][4], including the corresponding adaptation strategies. The test persons had to complete seven tasks on the user interface, while the simulated production environment always reconfigured between task 2 and task 3. The idea was that the adaptation should only affect task 4, which had to be executed in a different way (as a result of the adaptation) than indicated at first. The other tasks served as an indication that the test persons perform in a similar way. We conducted a *pilot study* with several computer science students. All six test persons were between 21 and 34 years old.

After being asked for personal statistical information, the test persons got a thorough introduction to the production industry domain. They were provided with a detailed explanation of the simulated production environment as well as of our Universal Control Device. After execution of task 2, we initiated the reconfiguration of the simulation and, depending on the strategy used (of which the test persons were not aware) the user interface adapted itself, notified the user, or demanded confirmation. The strategies were distributed equally between the tests. Due to the adaptation of the environment, the user should not be able to complete task 4 in the way the task description called for. One of three redundant pumps was removed from the system. Here, the user should conclude that (as displayed in the documentation of the simulated environment) there are various ways to complete this task and ask the moderator if this is possible. The test was completed with a survey about the subjective properties of the user interface.

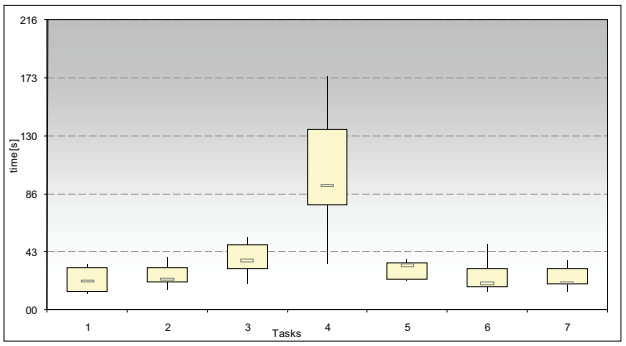

Figure 2. The performance of the 7 tasks (median, max/min and quantile).

Results

Figure 2 shows the results of the performance of the single tasks. As we assumed before conducting the experiment, all tasks (the control tasks) except task 4 were performed similarly. Referring to the standard deviation, which is an average of 16 seconds in case of the control tasks and 38 seconds in case of task 4, the difference in performance can be easily identified. Thus, we conclude that the results of this preliminary experiment are representative.

When executing task 4, the test persons recognized the redundancy of the three pumps and asked if they could use another pump, which was intended.

Discussion of the Results

All test persons were able to complete all tasks (with help in case of task 4), which confirms hypothesis H1. Since all of them needed help when executing task 4 and only during task 4, the estimated 85,6% of hypothesis H2 was almost exactly confirmed.

For ah-hoc adaptation, the execution of task 4 took an average of 117 seconds, for notification 125 seconds, and for confirmation 89 seconds. Hence, hypothesis H3 cannot be entirely confirmed, as ad-hoc adaptation outperformed notification, but still confirmation outperformed both of the other strategies.

Threats to validity

Because all the test persons were only computer science students, this may have had an effect on the result. But on the other hand, those students were already familiar with our previous work (without knowing the content of the experiment), which could also be a good simulation of domain knowledge. Nevertheless, the experiment needs to be conducted (and will be) using production industry domain experts. The most important threat to validity to mention is the fact that this was only the pilot study for the described experiment. This means that the sample was too small and therefore has no real statistical evidence, but it serves as a first indication as to whether an investigation according our idea would make sense.

CONCLUSION

When dealing with user interfaces in highly dynamic environments, such as intelligent production environments, there are special requirements. According to a shown dissonance in user interface design, when being applied in these environments, we have shown there is a special need for compensating usage errors. This can be achieved by systematically integrating adaptation strategies into the model-driven development process. Since multiple strategies exist, which provide different user experience, the performance with respect to our demonstrator was evaluated in a pilot study of a controlled experiment. First results show that there are differences in the performance and therefore some of our hypotheses could be verified.

ACKNOWLEDGMENTS

Our work as well as the GaBi project is funded in part by the German Research Foundation (DFG).

REFERENCES

1. Weiser, M. The computer for the 21st century. *Scientific American*, 265, 3 (1991), 94-104.
2. Breiner, K., Görlich, D., Maschino, O., and Meixner, G. Towards automatically interfacing application services integrated in an automated model based user interface generation process, *Proc. of the 4th International Workshop on Model-Driven Development of Advanced User Interfaces (MDDAUI)*, CEUR Workshop Proceedings Vol-439 (2009).
3. Breiner, K., Görlich, D., Maschino, O., Meixner, G., and Zühlke, D. Run-Time Adaptation of a Universal User Interface for Ambient Intelligent Production Environments. *Proc. of the 13th International Conference on Human-Computer Interaction (HCII-09)*, *LNCS Vol. 5613* (2009), 663-672.
4. Breiner, K., Görlich, D., Maschino, O., and Meixner, G. Automatische Generierung voll funktionsfähiger mobiler Bediensoftware aus Benutzungs- und Funktionsmodellen. *Proc. of Informatik, LNI Vol. P-154* (2009), 2210-2215.
5. Gould, J. D. and Lewis, C. 1985. Designing for usability: key principles and what designers think. *Communications of the ACM,* 28, 3 (1985), 300-311.
6. Nielsen, J. Ten Usability Heuristics. www.useit.com/papers/heuristic/heuristic_list.html.
7. Norcio, A. F., and Stanley, J. Adaptive Human - Computer Interfaces. *NRL Report 9148*, Naval Research Laboratory (1988).
8. Mitchell, J. and Shneiderman, B. Dynamic versus static menus: an exploratory comparison. *SIGCHI Bulletin,* 20, 4 (1989), 33-37.
9. Lee, L. and Johnson T. URCousin: Universal Remote Control User Interface. *Proc. of the Human Interface Technologies Conference* (2006).
10. Zuehlke, D.: SmartFactory – From Vision to Reality in Factory Technologies. Proc. of the 17th International Federation of Automatic Control (IFAC) World Congress (2008), 82-89.