User-Centric Adaptive Automation through Formal Reconfiguration of User Interface Models

Benjamin Weyers
Dept. of Computer Science and Applied Cognitive Science
University of Duisburg-Essen
Duisburg, Germany
weyers@inf.uni-due.de

Abstract—**This paper presents work in progress on a novel approach for modeling and implementing user-centric adaptive automation based on formal modeling and reconfiguration of user interface models. The approach addresses automation as relevant parameter in human-machine interaction; it is responsible for increasing workload during monitoring and control of complex technical systems and thereby for human errors in interaction. By actively involving the user into the adaptation process through user-side applied reconfiguration and system-side, workload depended adaptation, the user gets a deeper insight to the automation of the system and automation gets adapted to his or her needs. Thus, the main contribution of the work presented in this paper is the close integration of the user into the adaptation process of automation, resulting in a user-centric adaptive automation approach.**

Keywords-formal modeling; human-computer interaction; adaptive automation

I. Introduction and Motivation

In the course of the last decade, the increasing automation of complex controlling tasks has significantly changed how control of complex technical systems is done. Examples from the chemical and energy industry show that the main task of operators is to monitor the (automated) technical process rather than to control it [1]. Research in cognitive psychology has revealed important consequences of automation with respect to the human operator's workload in monitoring and control of technical processes, especially in critical, non-standard situations [2]. High workload is closely related to error rate, as well as to factors that influence the error rate in human-machine interaction, such as motivation, well-being, or situation awareness [3, 4].

Adaptive user interfaces are developed primarily in order to reduce workload and to increase human performance by adapting interaction to a specific user [5]. Thus, it seems obvious to adapt user interfaces in order to suit particular users' needs and to introduce into the adaption process the degree of automation as an important parameter influencing human factors in human-machine interaction [6]. Here, the degree of automation defines whether the user has more or less control over the process, which system information in a critical situation is provided, or how the granularity of input operations is defined. Still, adaptive user interfaces require information and data about the situation of user-system interaction to trigger and initialize certain adaptations. Here, mental workload has been identified as one primary reason for errors in user-system interaction. Especially in context of automated systems, the degree of automation is associated with potential increase of mental workload and thereby is an indicator whether the degree of automation is too high or too low and whether it should be adapted or not. Weert [7] describes how mental workload can be measured based on different physiological factors, such as heartbeat rate, facial expression, perspiration, or eye blink rate. Out of these factors, pupillometry has been identified as promising measurement for workload, especially in context of adaptive automation to increase human performance [8].

Motivated by these findings in cognitive psychology and work on adaptive automation, this paper presents an approach to combine system- and user-side triggered adaptation of the degree of automation as being integrated in a model-based and executable approach for describing user interfaces based on a formal and graph-based modeling approach for the creation of user interface models. This close integration of model-based creation and system-/user-side adaptation tries to make a first step into the direction of a "human-machine symbiosis", without losing the focus of creating computer-based systems [9]. This gap between modeling and implementation highlights the problem of transforming models into executable code, as has been described in context of software development [10].

Therefore, the model-based approach described in Section 2 is executable and offers mechanisms for model-intrinsic adaptation through graph transformation systems. Therefore, the presented approach is graph-based and is transformed into reference nets, a special type of Petri net. Still, computer-based adaptation of automation assumes the accessibility of automation in a system's architecture as a formal model or description. Based on this observation, Section 3 presents a modeling approach for integrating automation with the user's current workload into an adaption concept, targeting formal user interface reconfiguration using predefined adaption rules. These adaption rules getting instantiated based on the measured data and being applied in a next step to the formal user interface model. Finally, Section 4 will conclude the paper and will discuss future work aspects, such as a planned evaluation study

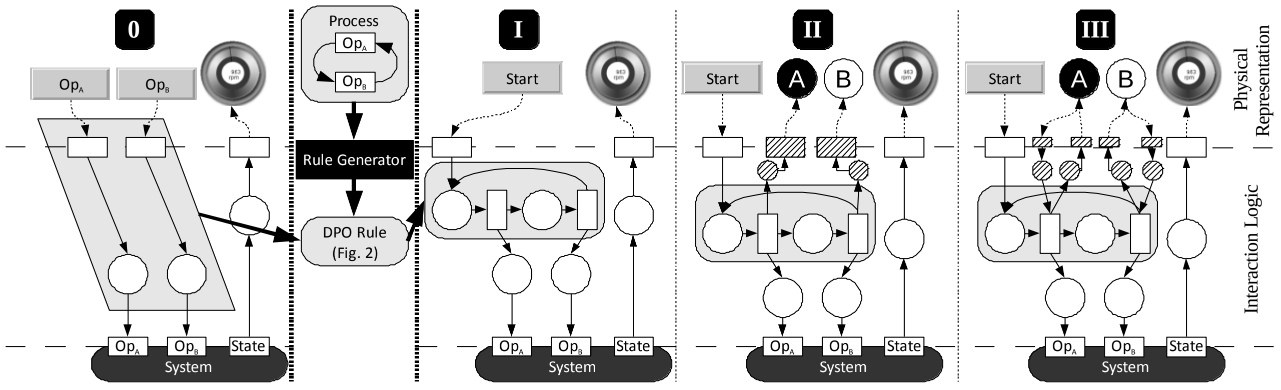

Figure 1. Example of a simple user interface reconfiguration to adapt automation.

investigating the influence of system-/user-side adaptation of automation to user's performance.

II. Formal Reconfiguration of User Interfaces

The adaptation of a user interface responding to the degree of automation, individualization for a certain user (or user group), as well as the necessity of the modeling approach to be executable, are the main arguments for the use of formal methods. This is because only fully formal models can be processed algorithmically in a computer-based system. For this reason, we developed a preliminary formal approach for modeling user interfaces in a two-layered architecture (see Figure 1, Section 0). The first layer represents the physical appearance of the user interface, which is directly accessible through the human user. Thus, it offers a set of input and output elements (button, slider, text fields, etc.) allowing the user to interact with the system and presents information (graphs, displays, instruments, etc.) to notify the user of the state of the system and any changes in it. The underlying second layer models the behavior of the user interface as a set of data processing routines that process events caused by the user and create certain data for controlling the system and, vice versa, process data sent by the system to be presented to the user. This two-layered architecture was used in various evaluation studies to reconfigure user interfaces (UIs) and to investigate the influence of reconfiguration to the human-computer interaction [11, 12].

For modeling the second layer, which is also called interaction logic, we developed a formal and visual modeling language called Formal Interaction Logic Language (FILL), accompanied by a visual modeling tool called UIEditor [13]. The UIEditor is able to model physical representation (such as a visual user interface), run a created user interface, and, finally, reconfigure the user interface based on a formal graph-transformation system. To execute a given user interface, interaction logic provided as a FILL graph is first transformed to a reference net, a special type of Petri net [14]. Reference nets provide formal semantics for FILL and makes interaction logic executable using Renew, a Java-based modeling and simulation tool for reference nets [15]. The simulation of a modeled user interface can be exemplary shown along the initial interaction logic (see Figure 1, Section 0). Here, the physical representation contains of two buttons labeled “Op_A” and “Op_B”, as well as an interaction element showing a single value on a certain scale, for instance, the current rounds per minute (rpm) of a pump. Thus, “Op_A” and “Op_B” could be operations to increase and decrease the rpm of the pump in the controlled system, which is represented as third layer (see Figure 1). The initial reference net-based interaction logic does nothing else than sending press events resulting from the user to the system and sending the current rpm value from the system back to the user interface’s physical representation. Therefore, press events of a button trigger transitions to fire in the interaction logic, as far as they are associated to that button (as it is indicated as dashed arrow in Figure 1). The same is true for the connection to the system and the callback mechanism for the rpm value.

To apply reconfiguration to a reference net-based interaction logic, transformation rules, which first have to be generated algorithmically, can be applied to the reference net and thus change the behavior of the user interface. The rules used in UIEditor’s reconfiguration component are based on the DPO (Double PushOut) graph rewriting approach [16]. An exemplary rule is shown in Figure 2. A DPO rule is divided into three different graphs, or Petri nets: The left side (L), the interface graph (I), and the right side (R) of the rule. The parts of the original graph G to be rewritten are defined by defining a matching function m. By applying the rule to

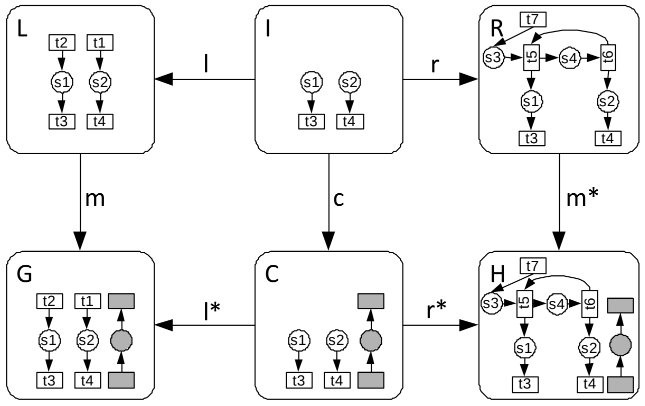

Figure 2. Double pushout (DPO) graph rewriting approach.

G, the difference between L and I is deleted from G (resulting in C), and the difference between I and R is added to C, yielding the final result, H. Figure 2 shows the rule that is applied to the initial interaction logic (see Figure 1**,** Section 0), where the result is shown in Figure 1, Section I, accompanied by a change of the physical representation. Here, a specific part of a control process (e.g., alternating increasing and decreasing of the pump's rpm) has been integrated into the interaction logic to automate the indicated process. Thus, automation becomes part of interaction logic as will be described in greater detail below. In many cases, this kind of reconfiguration of the user interface requires modifying the physical representation in respect to newly added interaction elements for input data or changes in how information is represented, as shown in Figure 1.

Thus, adaptation of user interfaces is implemented based on formal reconfiguration techniques that are used and integrated in a transformation-rule generator using context information, such as an automation model and user's current mental workload. The main contribution of using formal methods is that the model is still executable after creating and reconfiguring the user interface, where the whole adaptation process is implemented in the same formalism. This prevents solutions to be loose combination of different modeling approaches, which run the risk to be incompatible or losing information by changing between different modeling concepts [10]. The section below presents a coherent solution of adaptation of automation based on the former described self-contained modeling approach.

III. ADAPTATION OF AUTOMATION

For adaptive automation based on formal user interface modeling, it is assumed that the automation concept is fully accessible through an external formal model that matches the underlying concept of formal user interface modeling; therefore, the automation model should employ a Petri net–based representation. Based on this assumption, automation can be further understood as formal abstraction of interaction processes between the human user and any given system that has been technically implemented. Thus, in our sense, automation is part of the above-introduced interaction logic.

Assume a discrete and recurrent process of two operations "Op_A" and "Op_B", which have to be executed in iterative fashion, such as the process shown in Figure 1 and described before. According to the first assumption, this process can be introduced into reference net–based interaction logic as indicated by the bold arrow in Figure 1,

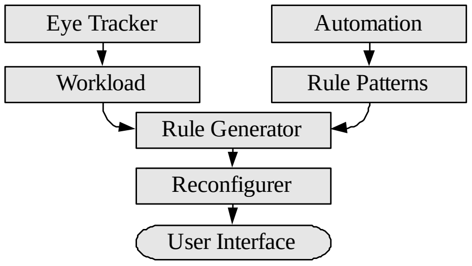

Figure 3. Conceptual architecture of the automation adaption module for the UIEditor.

which represents the application of the DPO rule shown in Figure 2. The automation of this process can then be started by the user pressing the newly added "Start" button. From this point on, the user is only able to monitor the system's state by observing the tachometer-like output widget, showing the current pump's rpm being controlled. Thus, using the initial interface, s/he is not able to follow the operations that are automatically executed by the interaction logic.

As Parasuraman [1, 2] described, workload increases during critical situations because the user has to understand the system's current situation, as well as how the automated control processes are reacting to the situation. The user has to gain insight into the automated process, resulting in increasing mental workload, sometimes dramatically. To adapt automation to this situation, the initial user interface can be reconfigured (see Figure 1**,** Section II), by adding more interaction elements providing deeper insight into the automated process. Two lamps are added to the physical representation accompanied by an extension of the interaction logic, now showing which operation is executed at any given moment. This makes interaction more finely grained and viewable to the user. A further reconfiguration extends the first by changing the simple lamps into buttons (see Figure 1, Section III), where the user is now able to control the automated process. Another possibility would be to remove the automation from the interaction logic and give all control back to the user without restriction or even, contrarily, to reduce the interaction and fully automate the process. The first would result in the initial user interface (see Figure 1, Section 0).

The second aspect for introducing adaptive automation into formal user interface models is that a system exists that is able to measure user's workload (beside other possible measurements) as s/he interacts with the technical system. Therefore, adaptive automation will be implemented as a component of the UIEditor framework, which combines the perceived mental workload of the user and his/er individual requirements. Here, three components have to be realized: the workload identification component, which observes the user through such means as an eye-tracker and calculates the workload from the captured data. The second component is the rule generator, which generates formal reconfiguration rules to be applied by the existing reconfiguration component in the UIEditor. The third component will be based on UIEditor's reconfiguration editor, which is able to apply individual reconfiguration operations to the interaction logic and thereby to the automation model by the user himself; called user-side adaptation above. Here, the user triggers the reconfiguration, selects the specific part of interaction logic to be reconfigured, and chooses the rule to be applied.

Various works have identified pupillometry as a possible indicator of user's workload [7, 8, 17]. Thus, the rule generator is triggered by the workload identification component to generate and apply reconfiguration to the user interface, what is called system-side adaptation (see Figure 3). Rules are then generated from rule patterns that have already been derived and described in automation research, resulting in a reconfiguration as shown in the above-described example, depending on the amount of detected workload (see Figure 1). Rule patterns consist primarily of specific structures to be added or deleted from reference net-based interaction logic and algorithmic implementation necessary to detect relevant parts of the reference net to be transformed. In conclusion, the combination of system- and user-side adaptation results in an integrative user-centric approach for adapting automation based on formal user interface models.

IV. CONCLUSION AND FUTURE WORK

In this paper, we presented a new approach to implement adaptive automation as part of a user interface modeling approach using formal methods to describe interaction logic. This integrative approach is a first step towards closing the gap between modeling, execution, and reconfiguration of user interfaces and integrating adaptive concepts to adapt user interface to user's needs, as well as making the degree of automation accessible for system- and user-side adaptation. Based on visual modeling and graph-based languages accompanied with algorithmic transformation of reference nets, user interface models get executable and can be reconfigured by graph rewriting systems. Considering automation as part of interaction logic, reconfiguration can also change automation beside further individualizations of the user interface concerning user's needs.

Finally, the described work in progress will be evaluated and further elaborated in context of former applied studies [12, 13]. Here, we investigated the influence of individualization of user interfaces using reconfiguration on monitoring and control of technical systems. The reduction of errors was in focus of these studies showing that it is possible to reduce errors through reconfiguration of user interfaces for interaction with technical system. Therefore, we will investigate the former introduced approach of adaptive automation concerning usability and applicability in a real monitoring and control scenario of a technical system (e.g., a simple simulation of a nuclear power plant), as well as analyze its influence on errors in interaction as measurable variable. We will run the test with two groups; one group will do predefined control tasks supported by system-side adaptation of automation, as well as user-side individual reconfiguration, whereas the control group will undertake the same tasks without support by the reconfiguration system.

The above introduced example (cf., Figure 1) is only one aspect of how automation fosters increasing workload and how it is possible to work against this effect through formal user interface reconfiguration. Therefore, future work will seek to identify further features of automation and to generate rule patterns in order to offer a comprehensive library that can deal with various situations and types of automation. Furthermore, an extension of the UIEditor tool for modeling, running, and reconfiguration via modularization and a more finely grained leveled architecture will be developed and implemented due to necessary refinement in the interaction logic model. Finally, the integration and reconfiguration of automation without necessarily including the user but other models and requirements will be part of future work.

REFERENCES

[1] R. Parasuraman, T. Sheridan, and C.D. Wickens, "A model for types and levels of human interaction with automation," IEEE Transactions on Systems, Man, and Cybernetics: Systems and Humans, vol. 30(3), 2000, pp. 286–297.

[2] R. Parasuraman and V. Riley, "Humans and automation: Use, misuse, disuse, abuse," Human Factors, vol. 39(2), 1997, pp. 230–253.

[3] C.D. Wickens and J.G. Hollands, Engineering psychology and human performance, Addison Wesley, 1999.

[4] M.R. Endsley, "Toward a theory of situation awareness in dynamic systems," Human Factors , vol. 37(1), 1995, pp. 32–64.

[5] A. Jameson, "Adaptive interfaces and agents," in Human-Computer Interaction Handbook, Erlbaum, 2003, pp. 305–330.

[6] R. Parasuraman, K.A. Cosenzo, and E. De Visser, "Adaptive automation for human supervision of multiple uninhabited vehicles: Effects on change detection, situation awareness, and mental workload," Military Psychology, vol. 21(2), 2009, pp. 270–297.

[7] J.C.M. Weert, "Ship operator workload assessment tool," Department of mathematics and computer science. Technical University Eindhoven, Eindhoven, 2006.

[8] T. de Greef, H. Lafeber, H. van Oostendorp, and J. Lindenberg, "Eye Movement as Indicators of Mental Workload to Trigger Adaptive Automation," in Proc. of Augmented Cognition, HCII 2009, LNAI 5638, Springer, 2009, pp. 219–228.

[9] P.A. Hancock et al., "Human-Automation Interaction Research: Past, Present, and Future," Ergonomics in Design: The Quarterly of Human Factors Applications, vol. 21(2), 2013, pp. 9–14.

[10] F. Heidenreich, J. Johannes, M. Seifert, and C. Wende, "Closing the Gap between Modelling and Java," Software Language Engineering, LNCS 5969, Springer, 2010, pp 374–383

[11] B. Weyers, D. Burkolter, A. Kluge, and W. Luther, "Formal modeling and reconfiguration of user interfaces for reduction of human error in failure handling of complex systems," Human Computer Interaction, vol. 28(10), 2012, pp. 646–665.

[12] B. Weyers, W. Luther, and N. Baloian, "Interface creation and redesign techniques in collaborative learning scenarios," Future Generation Computer Systems, vol. 27(1), 2011, pp. 127–138.

[13] B. Weyers, Reconfiguration of user interface models for monitoring and control of human-computer systems, Dr.-Hut, 2012. http://www.uieditor.org/publication.html

[14] O. Kummer, Referenznetze, Logos, 2002.

[15] O. Kummer et al., "An extensible editor and simulation engine for Petri nets: Renew," Applications and Theory of Petri nets, LNCS 3099, Springer, 2004, pp. 484–493.

[16] H. Ehrig, K. Hoffmann, and J. Padberg, "Transformation of Petri nets," Electronic Notes in Theoretical Computer Science, vol. 148(1), 2006, pp. 151–172.

[17] T. Halverson, J. Estepp, J. Christensen, and J. Monnin, "Classifying workload with eye movements in a complex task," in Proc. HFES Annual Meeting, Human Factors and Ergonomics Society, 2012, pp. 168–172.