Rapid Development and Evaluation of Interactive Systems

Jörg Marrenbach
Department of Technical Computer Science
Aachen University of Technology, Germany
Phone: +49-241-44026105 Fax: +49-241-8888308
Ahornstrasse 55, D-52074 Aachen
E-Mail: marrenbach@techinfo.rwth-aachen.de

Abstract – The interaction between humans and machines is ubiquitous in daily life. Some interaction techniques are more efficient or effective than others for certain tasks or users. During the development of new systems, the user is referred frequently too late into the development process. This is due to the fact that a prototype does not exist to verify the system by the user.
This article outlines some ideas regarding the relationship between the systems specification and the user requirements at this early stage of the design process through the use of an appropriate specification tool. This tool has been developed by the Department of Technical Computer Science that enables the designer to create models of tasks, users, and devices and to connect these models to visualise the relations between them.

1 Introduction

The interaction between humans and machines is ubiquitous in daily life. Some interaction techniques are more efficient or effective than others for certain tasks or users.
The traditional methodology within the development of software-based products was essentially based on a linear, phase-oriented life cycle. This approach is also called “waterfall” model. It consists of different levels for definition of requirements, specification, design, implementation, validation, verification, operation and maintenance. Such a model presupposes that customers or end users are able to formulate their conceptions very precisely at the beginning of the development process. This is not the case according to experience. The requirements are reflected to shorten the development process, to overlap the individual phases and enable continuous modifications.
This led from the linear to the iterative model of software development, in which in early phases creating of prototypes is possible already. ISO 13407 [1] describes such an user and task oriented development cycle (Fig. 1).

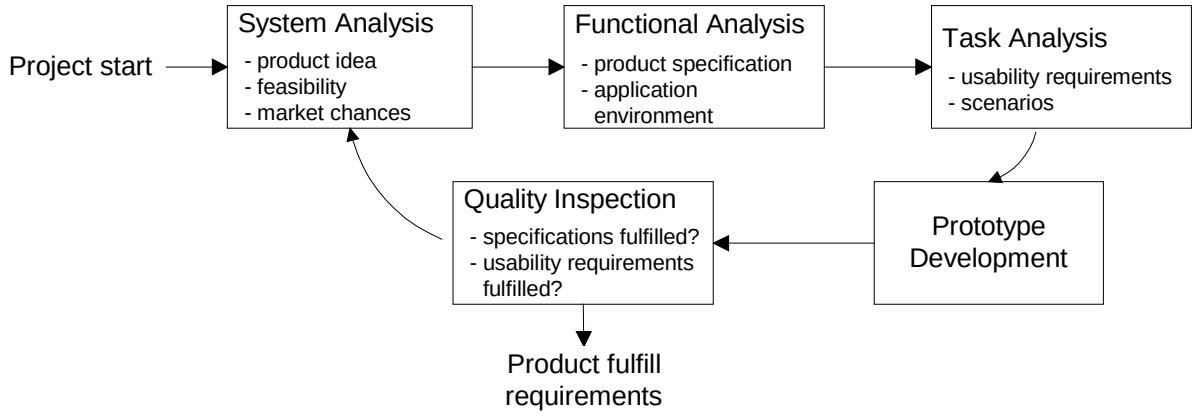

Fig. 1: Development cycle for interactive systems

First of all, the user requirements, their functions and the intended application environment are to be determined. From this analysis the usability requirements can be derived and from these results first product designs (prototypes) can be developed in a third step. These prototypes have to be verified for agreement with the specified usability requirements. Also end users must be queried. After the identification of further development needs the cycle starts again, until all requirements are fulfilled.

2 Motivation for a new Methodology

As system complexity increases, it becomes more difficult to completely specify detailed requirements in text form. The documents that attempt to describe these systems become large and complex. The requirements may interact in intricate and complex ways. The review and sign-off processes can be lengthy and expensive. Verifying that the requirements documentation is complete, accurate and consistent can be a daunting or even impossible task.
As the problems to be addressed increase the complexity, the solution approaches become less obvious. Software developers will not be experts in the domain of the problems to be solved, so it is similarly unrealistic to depend solely on them to define a system. A cooperative effort, among domain experts and technology experts, to discover system requirements can leverage the value added of new systems. Ultimately, the need for a better way to develop software systems is driven by the need to manage the risks involved, i.e., development costs.

3 Concept for Modern Software Development Environment

While many system development efforts still claim to use the waterfall model, in the trenches programmers, analysts and project managers are devising more effective techniques.
This section outlines a concept regarding the relationship between the system specifications and the user requirements at an early stage of the design process. The concept can make effective use of tools to integrate the requirements analysis, design, code and test environments. This concept enables the developer to create models of tasks, users and devices and to connect these models to visualise the relations between them. Based on these models an analytical evaluation of the system can be done. Additionally, high quality systems can be developed much faster using this concept as opposed to the waterfall approach, especially by using the different models. Furthermore, it can be assumed, that user satisfaction with the systems developed improves when using this concept.
While there is a wide range of tools for specifying device models and task models for more or less complex devices, no tool seems to offer an easy way to create an accompanied user model to a device prototype. Therefore in most cases, the first stage at which the designer can determine the quality of human machine interaction is when the device is fully built and real user studies are conducted. However, it would be much more convenient to build a normative user model as part of the device specification, and having a tool to evaluate the prototype.

4 System Architecture

For the evaluation of man-machine interfaces regarding the adequacy three criteria are considered as essential. According to ISO 9241 part 11 [2] these are named effectiveness, efficiency and satisfaction. These criteria determine the general evaluation frameworks for the investigation of man-machine systems. In addition, further specification demands are considered, as specified, e.g., in ISO 9241 part 10 [3].
With consideration of the above-mentioned criteria an environment is developed, which enables the developer to evaluate a technical system promptly regarding to ergonomic requirements. Figure 2 shows the dependencies of the development environment. On the basis of a

functionality already specified in the requirements a device model is created. Subsequently a user model as well as a user interface are built up.

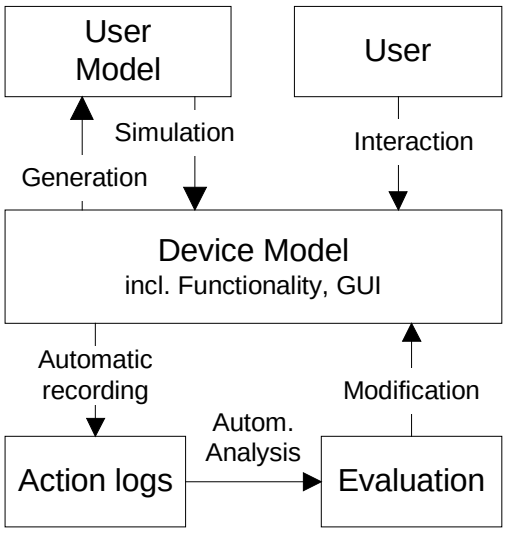

Fig 2: System architecture of the development environment

The evaluation can be done by a simulated use via the user model and by test with prospective users. Action logs are generated automatically and conclusions concerning the operability of the prototype can be drawn, e.g., from errors and learning times.

4.1 Device Specification

A usual device prototype consists of several connected and dependent models. The device model contains details about the inner works of the device, specified as states and transitions. For the specification of the device model statecharts are used [4]. Statecharts are an extension of usual transition diagrams, offering the possibility for hierarchical and concurrent state organisation [5]. A device is modelled as a set of states interconnected by action transitions. A task model describes what a user is expected to do with the device. This model is created by the systems designer and can be seen as an early form of the owners manual of the device [6].

4.2 User Modelling

The user model, while similar to the task model in its relation to user actions, is extended by specific values for the user workload and complexity of tasks. User models can be used for different purposes. While they are no integral part of a device, user models can help making devices easier to use. Special adaptive user models can even support the user actively in interacting with the device. For designing and prototyping matters, normative user models are of interest. They describe a normal user with average knowledge about the devices and its purpose. Contained in the normative model is a hierarchy of operations to be performed in order to achieve a goal [7].

5 StateWatch Approach

The tool currently under development, called *StateWatch*, is used to specify the device behaviour in form of graphical data diagrams (statecharts), to automatically generate source code from these diagrams and to run in real-time. Additionally, a graphical user interface (GUI) is created from the device model's states and actions, and the system designer is able to define a user model and add hints for task complexity and performance measurement. The created software prototype is executed directly from the tool, without writing or generating

code manually, to see if it fits the users requirements. Action protocols are logged during test sessions automatically. The transitions of states is displayed graphically to allow the designer a better validation of the device model.
Furthermore, the normative user model created as part of the software prototype is able to rule the user interface elements, causing actions a test user is expected to perform. Protocols logged during these automated test sessions can afterwards be compared to real user logs, allowing for an easier evaluation.

5.1 Used Tools

Statemate is the application used to create statecharts for transition diagrams that specify the device behaviour [8]. Statemate is able to build up executable software versions of the modelled devices, however a real GUI creator is missing.
The market offers various products for designing user interfaces, but only some are able to simulate the created user interface directly without writing, generating, or compiling code. The GUI tool that is used as a basis for user model integration is called *InterfaceBuilder* [9]. StateWatch can be implemented as a palette extension to InterfaceBuilder. Several new interaction and control elements are located on the palette and can be integrated into a device prototype.

5.2 Technical Aspects

InterfaceBuilder is a tool that works in a object oriented way. The developer creates an application by dragging objects from palettes, arranging them on a workspace, and wiring them up graphically. StateWatch adds objects to this environment. There are common GUI objects for control lights, windows, buttons, etc. with somewhat changed behaviour compared to the default elements. In addition, objects for connection to Statemate (ActionWrapper) and for creating and editing a user model are available. Figure 3 depicts StateWatch as a palette extension to InterfaceBuilder.

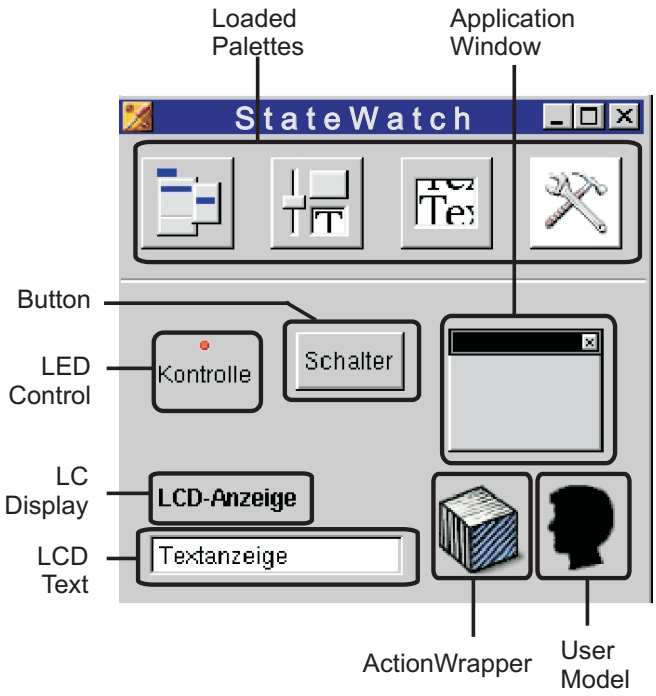

Fig. 3: StateWatch integrated into InterfaceBuilder

StateWatch requires a device model to be created with Statemate, specifying the device behaviour and possible user actions. The system designer can then create a GUI using the full InterfaceBuilder functionality, connecting the interface elements back to actions from the statechart. Output elements, called data sinks, can be connected graphically to data sources like other GUI elements or states.
The task analysis following the GOMS approach, a formal description technique [10, 11], is used for building the user model. StateWatch enhances the GOMS modelling technique in that way that complexity information can be added to operators. In that way the total complexity to achieve a goal can be determined during a test run the user model performs.

5.2 Application

A CD player prototype was created to test the StateWatch features. In general, all CD players are quite similar in use and it is save that to assume that the most people are familiar with their usage. Figure 4 shows the devices statechart modelled with Statemate and the related graphical user interface of the prototype which is created by using InterfaceBuilder.
Only a small set of the device functionality are considered. However, with these functions the usage of the prototype can be described in general.

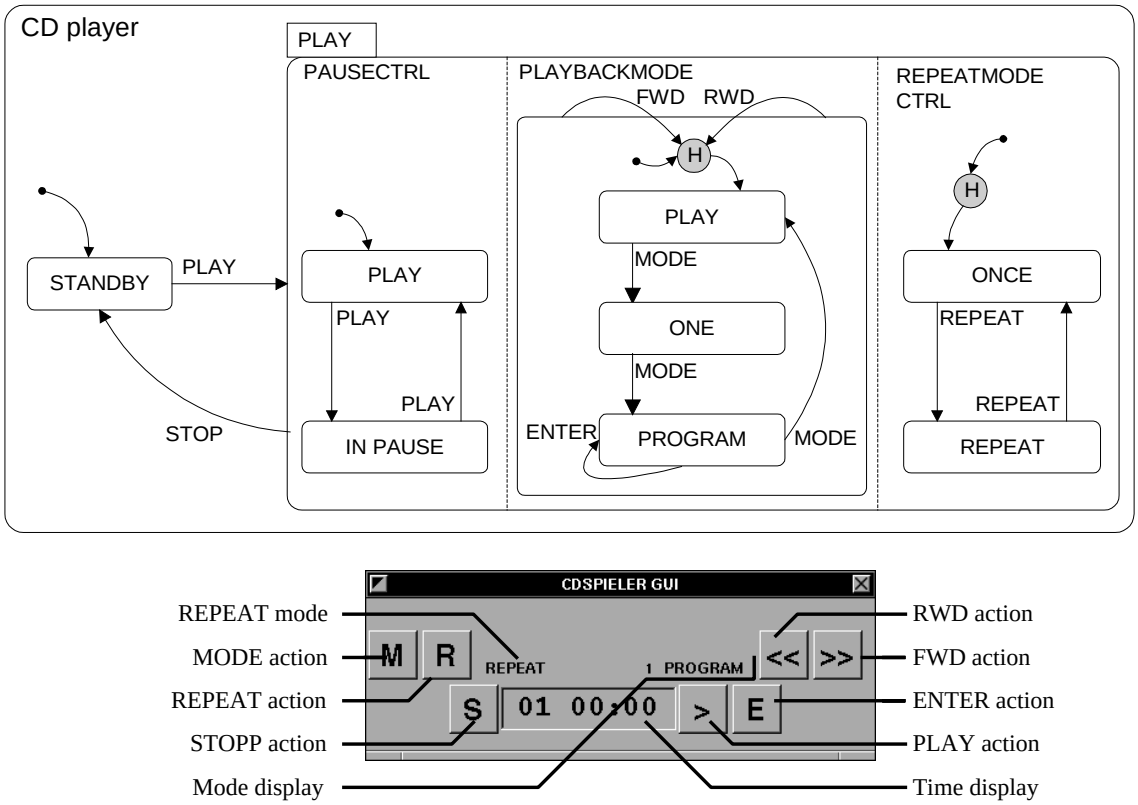

Fig. 4: Statechart modelled with Statemate and the related user interface of the CD player

The statechart models the individual states of the CD player and the possible transitions. Within the playback mode three different modes exist (PLAY, ONE , PROGRAM). Independently of these modes it can be selected between simple (ONCE) and repeated playback (REPEAT). A further parallel state controls the pause mode. The buttons on the user interface are marked with the initial letters of the transitions (STOP, ENTER, MODE and REPEAT) as specified in the statechart. The internal messages FWD and RWD are implemented not in the statechart, but in functionality.

6 Conclusions and future work

In this work the development environment StateWatch is presented, which consists of available tools and developed extensions. The environment offers support according to the task analysis and the specification of functionality within the development of the dialog control and a graphical user interface as well as a normative user model. The result is an executable prototype. The operation can be simulated by the normative user model.

As a special feature of this environment can be outlined that the development of the different models can be done without editing program code. The dialog control is specified by Statecharts in a graphical editor. The graphical user interface can be designed with InterfaceBuilder. The development and modification of a user model occurs with an editor, that is integrated in InterfaceBuilder. The goal of StateWatch is to show that an integration of normative user models can simplify the evaluation of technical systems.

In the future some extensions of the system are planned. These concern the further automation of the development process. One point is the automatic generation of the user model from the interaction specification of the Statecharts.

7 References

[1] ISO 13407 - Draft: *User centred design process for interactive systems*. International Organisation for Standardisation, Genf, 1998.

[2] ISO 9241-11: *Ergonomic requirements for office work with visual display terminals - Guidance on usability*. International Organisation for Standardisation, Genf, 1997.

[3] ISO 9241-10: *Ergonomic requirements for office work with visual display terminals - Part 10: Dialogue principles*. International Organisation for Standardisation, Genf, 1996.

[4] Marrenbach, J: *Modelling Complex Systems using Statecharts applied to the User Interface of an Advanced Flight Management System*. In: Proc. of the IEEE International Conference on Intelligent Engineering Systems INES´98, Vienna, Austria, pp. 43-48.

[5] Harel, D.: *Statecharts: A Visual Formalism for Complex Systems*. In: Science of Computer Programming, North Holland, Elsevier Science Publishers, Vol. 8, pp. 231–274.

[6] Kraiss, K.F.: *Modellierung von Mensch-Maschine Systemen*. In: Verläßlichkeit von Mensch-Maschine Systemen. ZMMS-Spektrum, Vol. 1, Berlin, 1995, pp. 15-35.

[7] Marrenbach, J.; Leuker, S.: *Konzept zur Evaluierung technischer Systeme in der Entwicklungsphase*. In: ITG-Fachbericht, Vol. 154, 1998, pp. 103-112.

[8] i-Logix Inc.: *Statemate Magnum - User Guide*. Andover, MA, 1997

[9] Apple Computer, Inc.: *Discovering OPENSTEP: A Developer Tutorial*. 1997.

[10] Card, S.; Moran, T.; Newell, A.: *The psychology of human computer interaction*. Lawrence Erlbaum, 1983.

[11] Kieras, D.; Polson, P.: *An approach to the formal analysis of user complexity*. In: International Journal of Man-Machine Studies, Vol. 22, 1985, pp. 365-394.

[12] Irving, S.; Polson, P.; Irving, J.: *A GOMS Analysis of the Advanced Automated Cockpit*. In: Proceedings of CHI ´94, 1994, pp. 344-350.