# Towards Automation of Usability Studies

Björn Schuller, Frank Althoff, Gregor McGlaun, Manfred Lang, Gerhard Rigoll
Technical University of Munich
Institute for Human-Machine-Interaction
Arcisstraße 16, D-80290 Munich, Germany
(schuller | althoff | mcglaun | lang | rigoll)@ei.tum.de

## I. ABSTRACT

This paper describes partial automation of Wizard of Oz usability studies. Such simulations allow for interactive usability testing and user feedback on software or human-machine-interfaces prior to building a working prototype. We believe that the wizard needs tool support to be effective and have developed a tool called UsaWiz to help automate the interpretation of generic multimedia user input cues, and in quickly developing high level prototypes. In particular, this concept addresses one of the weaknesses of the major usability engineering techniques: the reproducibility of the study under the same conditions. The subjective influences of test conductors are limited to a minimum. Furthermore all test-data is streamed efficiently. The introduction of a usability experiment control language and a modular unit construction system are the heart of the generic nature of the concept. Besides the basic goals a realization of the presented ideas is introduced and evaluated regarding in particular performance and savings achieved by the approach. Especially for studies in the automotive environment an exemplary distraction task and baseline evaluation are also explained in detail.

*Keywords*: **Usability Engineering, Wizard of Oz Studies, Computer Based Usability Engineering, Automation of User Studies**

## II. INTRODUCTION

The Wizard of Oz principle is a far known technique to evaluate a user interface concept or attain real-world data. It is based on the idea of simulating a fully working system by a human conductor called "wizard" without the need of developing functional prototypes [1]. The probands on the other hand experiences a perfect working application. This allows for early iterative redesign by showing the actual usability of a concept. However, a disadvantage of the technique is the wizard's personal influence conflicting with the basic condition of reproducibility of a study. Such interferences are among others alternating wording for the test-instructions, diverging reaction times, or simulation errors. In particular studies for the evaluation of multimodal user input like hand, pen or head gestures, speech and emotion interpretation are in the scope of nowadays research [2]. Wizard of Oz experiments are often the basis of a data collection to train stochastic multimodal human input engines. Such recognition modules in general underlie recognition errors, which are simulated in conceptual studies. However, shortcomings of a test conductor are not analogical to real occurring recognition errors. Nevertheless it is often essential to simulate realistic recognition errors what seems to be a hard task for a test-conductor. Furthermore given a complex user interface structure a high level of cognitive workload is required for modeling the complete test-system structure mentally as a human. The required attention to manage the flow of tasks prevents observation of the ongoing study and the test person. Like this in conventional Wizard of Oz studies data must be completely reviewed in a time-consuming process after a study. Before a test can be started, wizard training is in general required. This demands time and experience, and often results in the loss of test-persons by serious early wizard faults.

These deficits lead to the idea to aid a test conductor by a generic automation concept [3]. In the field of computer based usability engineering (CBUE) first specialized systems have been introduced [4][5] to accomplish single tasks in a semi-automated concept. We introduce an integrative and generic system called *"UsaWiz"*. Computer aid manages synchronization in exact timing of different tasks as required for example in an automotive simulation when testing an interface function in a driving situation. By an initial baseline measurement [6] the test environment can be adjusted to individual skills of probands, instructions and questions are played, the wizard is assisted and all data is integrated and analyzed in a log file automatically.

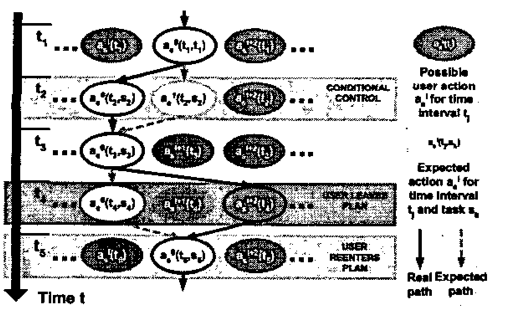
Finally this method also allows the integration of systematic errors for the simulation of real occurring recognition errors. Also the principle of reproducibility is preserved by this opportunity.

## III. Principle

### A. Probabilistic Predictability of Usability Experiments

The basic idea is that an experiment often follows a previously known storyboard. This scheme describes the general flow of tasks which a user should fulfill and the according reactions of the simulated system. This limits the number of possible user actions at a time to the most probable in the actual context and schedules a study reproducibly. A storyboard can now be transcribed in a language designed for automation.

To give an example we assume the test of an audio device in an automotive environment controlled by natural speech and further modalities. There are 20 basic user actions and different parameters giving an exponential number of combinations. The actual task is designed to evaluate the concept of track selection. The test person is instructed to skip to a concrete track and utters: *"Please skip to track x performed by artist y on album z"*. The fact, that this interaction is expected after the announcement of the task, enables our system to provide the according action instantly. A wizard in normal set-ups would have to chose the right album and track and start a player to simulate a perfect working system. This normally demands a multiple of time.



Time t
CONDITIONAL CONTROL
USER LEAVES PLAN
USER REENTERS PLAN
Possible user action a_s^i for time interval t_j
Expected action a_e^i for time interval t_j and task s_k
Real path
Expected path

*Figure 1: Principle of predictability*

Generally, if we know the task $\tau_k$ demanded at time $t_j$ we can limit the total number $N_{\alpha s}(t_j)$ of currently expected system interactions $\alpha_s^i(t_j)$ to a fraction $N_{\alpha e}(t_j,\tau_k) << N_{\alpha s}(t_j)$ of a-priori expected interactions $\alpha_e^i(t_j,\tau_k)$ depending on the task. The expected user actions can be provided to the wizard via a graphical interface instead of the enforcement to handle the full functionality of the test interface manually as in common test environments. This in general highly reduces the wizard reaction time $T_{wr}(N_{\alpha s})$ to $T_{wr}(N_{\alpha e})$. If the user action $\alpha_u(t_j)$ differs from the estimated actions $\alpha_e^i(t_j,\tau_k)$ the wizard can nevertheless switch to manual control, with the only intention to lead the test-person back to the original plan by-passing the loss of simulation. Generally the test can still be seen as reproducible by neglecting such lost sequences.

A further gain can be achieved by probabilistic evaluation of the selected actions of first or higher order throughout the ongoing tests. In antecedent cycles the system may additionally sort or highlight more probable actions. Also this visualizes first results in a very early test-phase.

### B. Experiment Control Language

With respect to the mentioned points we introduce an experiment control language, *"XCL"* [7]. Its abstract syntax provides features as communication control between different task units, conditional control structures, loops, timing routines in a millisecond resolution, jumps, and, finally special markers and remarks for the wizard, and the analysis of the test results.

```
<S>               ::=  <CMD_SEQ>
<CMD_SEQ>         ::=  <CMD_LINE> |
                       <CMD_LINE> <CMD_SEQ>
<CMD_LINE>        ::=  <TASK_DEF> |<TU_CMD> |
                       <TS_CMD> | <GUI_CMD> |
                       <WAIT_CMD> |
                       <JUMP_CMD> |
                       <REACT_CMD> |
                       <WIZMSG_CMD>
<TASK_DEF>        ::=  task <TASK_ID>
                       <TASK_DESC>
<TASK_ID>         ::=  <chars>
<TASK_DESC>       ::=  <chars>
<TU_CMD>          ::=  sim <tu_command> | tu
                       <TU_ID> <tu_command>
<TS_CMD>          ::=  app <ts_command>| ts
                       <TS_ID> <ts_command>
<GUI_CMD>         ::=  gui <gui_command> | gui
                       <GUI_ID> <gui_command>
<WAIT_CMD>        ::=  wait <digits>
<WIZMSG_CMD>      ::=  wizmsg <chars>
<REACT_CMD>       ::=  <WIZWAIT_SCR> |
                       <WIZWAIT_CMD>
<WIZWAIT_SCR>     ::=  wizwait_script |
                       wizwait_script <rcscript> | ...
<WIZWAIT_CMD>     ::=  wizwait_cmd |
                       wizwait_cmd
                       <INTERN_CMD> | ...
<INTERN_CMD>      ::=  <WAIT_CMD> | <TU_CMD>
                       | <TS_CMD> | <GUI_CMD> |
                       <WIZMSG_CMD> |
                       <JUMP_CMD> |
                       <SCRIPT_CMD>
<JUMP_CMD>        ::=  skip_task | repeat_task
<SCRIPT_CMD>      ::=  script <rcscript>
<WIZMSG_CMD>      ::=  wizmsg <chars>
```

| | | |
|---|---|---|
| **<digits>** | = | numeric charset |
| **<chars>** | = | alphanumeric charset |
| **<rcscript>** | = | run-chart filename |
| **<tu_command>** | = | cfg-command of a task unit |
| **<ts_command>** | = | cfg-command of test-system |
| **<gui_command>** | = | cfg-command of test-wizard GUI |

*Figure 2: Excerpt of XCL in Backus-Naur-Form*

The language is interpreted, giving the possibility of immediate testing. An XCL-script will be called *run chart* in the following.

### C. System Architecture

The XCL-interpreter, called *control unit*, pilots the test system, as well as different so called *task units*. The idea of modular task units leads to a generic approach. Different tasks such as prerecorded wizard speeches, automatic questionnaires for the test person or even the wizard or especially designed attention tasks for distraction tests can be integrated in the basic concept. Each task unit possesses its own application specific commands wrapped in a context free grammar formalism allowing for distribution on heterogeneous platforms. The communication between the units is realized via Internet socket communication enabling remote control.

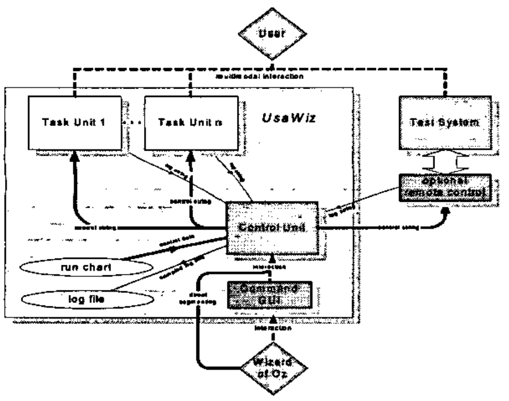


*Figure 3: System overview*

### D. Role of the test conductor

A study is split in sub tasks. These can be repeated or skipped and the task announcements can be replayed. Generally the wizard waits for a user-action and validates it only with the expected actions. The UsaWiz system helps the wizard online with hints and activates a wizard questionnaire to achieve an expert's summary already in the ongoing test. Basically the main work is shifted to preparation. However, programming a study is in any case less effort than wizard training.

### E. Exemplary Task Unit: Distraction Tasks

As an example of a task unit a distraction task for studies in the car environment is chosen. Evaluating user interfaces in automotive environments in non-field experiments demands a driving simulation for realistic circumstances. For the main goal of our methods is as mentioned to preserve reproducibility, also a driving simulation has to be designed controllable for particular requirements. The next figure shows a track situation designed for a parallel operation task with the test interface.

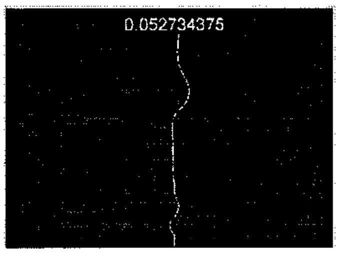


*Figure 4: Driving line adapted to operations tasks*

In the following figure parameters of the driving task can be seen. These can be used to adjust the setting or to preserve the user's motivation by changing scenes.

| Group | Variable |
|---|---|
| **Car** | Steering sensitivity, top speed, acceleration, reaction contour of brakes, vehicle inertia |
| **Area of view** | x, y, and z limits of visible area |
| **Tracking** | Vehicle width, lane width, total lane number, obstacle parameters |
| **Visualization** | Perspective distraction, shape of lane markers and road signs, season, illumination |

*Figure 5: Parameters of the driving task*

Yet, it often showed that a set of participants had nearly no difficulty with the driving task, whereas others were demanded so much that they were hardly able to operate the in-car application. Subjects of the latter group showed the tendency to no longer take the driving task seriously, and solely concentrated on the operation of the in-car application. To avoid these unrealistic scenarios and keep the focus on security, the design of this kind of trials requires a rather sensitive taxonomy of the specific influence factors.

## *F. Measuring the performance in Distraction Tasks*

In this chapter the example is extended to explain the principle of baseline measurement. To ensure an identical driving workload for each test subject, it becomes necessary to individually adapt the degree of difficulty (DOD) of the driving task. We developed a technique in which concerning the driving performance, each test participant is pre-classified in a baseline investigation before the main trial.

Before the main test, each test person participates in a so-called baseline investigation in which the subject is pre-classified with regard to his driving performance. Within the baseline investigation, which runs over a period of 180s, the participant is not yet confronted with any kind of operation task. The subject has to follow a predefined course of a road that is unknown to her or him. For reproducibility reasons, in this ascertainment, the participant cannot manipulate the velocity, but by a predefined run chart, the speed of the car is varied automatically between five DODs. The degrees are characterized by the frequency, the narrowness of the curves, and the speed. In general, the time slots in the sequence of the DODs are of different length, and the levels are not monotonously de- or increasing to eliminate anticipation effects by the test persons. The requirement throughout a driving task is to always keep the car on the right lane of the street. Please consider figure 5 for the identifiers used in the following. Technically, the vertex of the triangle $T$ (representing the current position of the car) must be kept within the limits (distance $q$) given by the broken white line $A$ in the middle of the road and the right road side $B$ .

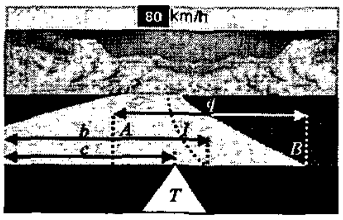


*Figure 5: Task Unit example Driving Task*

During the whole study, the abscissa $c$ of the position of the vertex of $T$ in relation to the abscissa $b$ of the ideal line $I$ and the correspondent timestamps $t$ are recorded in a log file (at a sample rate of 13Hz). Directly after the baseline investigation, the test subject is pre-classified according to her perfomance, and, based on the result, the DOD to be used in the driving task of the main trial is determined. If a participant had a bad performance in the baseline investigation, the DOD will be respectively low in the main part and vice versa. To objectively rate the driving performance of each test subject, a special numerical validation measurement is introduced, as follows:

If the driver leaves the right lane of the road, i.e. in case *the condition* $| c - b | - 0.5q < 0$ *is violated, a so-called deviation event* $D_j$ is generated. Then, $d = | c - b | - 0.5q$ represents the distance of the car from the boundary of the lane in the simulation. Let $j$ be a counting index for the deviation events, $i$ for the distance samples, $m(j) = |D_j|$, and $d_{1,1}$ denote the first recorded distance of the first deviation event $D_1$. Then, a deviation event $D_j$ can be characterized by a set of distance samples, as follows: $D_j = \{d_{1,j}, d_{2,j}, \ldots, d_{m(j),j}\}$. Each deviation event is validated with a numerical value $E_j$. *This measurement is dependent on the* individual performance factors:

i) the current distance $d_{ij}$ at the timestamp $t_{ij}$ and
ii) the DOD $s_{ij}$, at the timestamp $t_{ij}$.

The $j$-th contribution $E_j$ of one deviation event $D_j$ is now computed as the linear approximation of the area between the edge of the lane (A, and B, respectively) and the driven line. Assumed the car has exceeded the *left border $A$, in figure 2, an exemplary visualization for* the computation of $E_j$ is given (bird view, scene rotated by -90°):

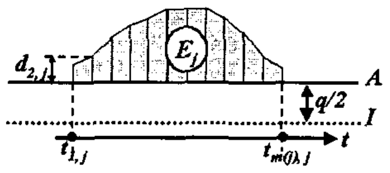


*Figure 6: Bird view of -90° rotated scene*

Thus, $E_j$ can be computed via the following formula:

$$E_j = \sum_{i=2}^{m(j)} \frac{(t_{i,j} - t_{i-1,j})(d_{i,j} + d_{i-1,j})}{2}.$$

Assumed $n$ deviation events, the total error score $E^*$ is calculated as the weighted sum of all $n$ score contributions. The weight of each single deviation is determined by $s_{1,j}$ which is the DOD having been effective at the time $t_{1,j}$:

$$E^* = \sum_{j=1}^{n} \frac{E_j}{s_{1,j}} = \sum_{j=1}^{n} \sum_{i=2}^{m(j)} \frac{(t_{i,j} - t_{i-1,j})(d_{i,j} + d_{i-1,j})}{2s_{1,j}}.$$

The score $E^*$ is compared to the values of a LUT, that was determined in a preliminary study to be described elsewhere.

ancillary conditions: resolution 1024x768 pixels, $r$ = 1pixel (car represented by vertex of a triangle), $q$ = 380 pixels, DOD 1 = easy,…, DOD 5 = hard:

| *a) score $E^*$ (in ms pixels)* | 0-3000 | 3001-6000 | 6001-9000 | 9001-12000 | 12001- |
|---|---|---|---|---|---|
| *b) Level* | 5 | 4 | 3 | 2 | 1 |

*Figure 7: Levels for the base-line evaluation*

The principles introduced may also be used in other domains. Examples for further distraction tasks might be office-tasks like typing for the evaluation of a new telephone concept.

### G. *LaunchPad*

The individual components of the UsaWiz test environment are coordinated by a special module called *LaunchPad*. This tool represents the primary interaction front-end for the test conductor to first set up the tool set and second to control the appropriate modules. For exchanging information between the individual modules of the system we are using an extended context-free grammar formalism [10]. In a classical client-server approach, string messages are exchanged over TCP/IP sockets.

The LaunchPad offers various functionalities. Individual components can be started and stopped by simple button clicks. For a coarse online analysis, the message flow can simply be observed by an additional log window. Moreover, the LaunchPad offers a text console to quickly react to certain unpredictable situations. The entire set of parameters for the individual system components is stored in the form of ASCII text files, that can easily be edited and, thus, changed very conveniently. This feature extremely reduces the complete set-up and reconfiguration time that would normally be necessary. The LaunchPad can be used in a heterogeneous system environment. To enable a broad control of potential modules, we have realized a wide range of front-ends, including binaries on various system platforms, Tcl/TK, Perl modules and Java applets.

## IV. Results and Conclusions

The system itself was used in several studies. It proved that the wizard's cognitive workload could be reduced to a fraction. The following table shows two examples of the achieved relative reduction rate $\rho_\alpha$, the in average provided commands $\tilde{N}_{\alpha}$, and the speed-up factor $\sigma_T$ built by the mean wizard execution time $\tilde{T}_w(\tilde{N}_\alpha)$ in multimodal studies testing a car-interface [8] and interaction in virtual 3D-worlds [9]. The first value shows the average number of addressable functions $\tilde{N}_\alpha$ and in round brackets of complex shortcut commands including associated parameters. In these studies the expectation path loss rate was in average 4%, but all losses could be caught. The gained speed-up enabled us to simulate even real-time processing future input modalities.

| | **Sys. actions** $\tilde{N}_\alpha$ **(plus param.)** | **Sys. action reduction** $\rho_\alpha = \frac{\tilde{N}_{\alpha}}{\tilde{\tilde{N}}_\alpha}$ | **Exec. time speed-up** $\sigma_T = \frac{\tilde{T}_w(\tilde{N}_\alpha)}{\tilde{\tilde{T}}_w(\tilde{N}_\alpha)}$ |
|---|---|---|---|
| **Car-MMI** | 20 (>>$10^4$) | 7% | 13 |
| **3D-Worlds** | 116 (>$10^3$) | 2% | 21 |

*Figure 8: Table of reduction achievements*

Due to the remark functions the XCL-script can be used as precise description of a study for a databank. The log data can be directly used for session replays. We believe that the work presented is an important contribution to all the effort towards a computer-based usability engineering environment. A new automatic approach to correct interferences during Wizard of Oz usability studies could be provided. The ideas presented can be transferred into other domains as software or web-design testing. However, if possible expected user actions are broad the system loses its potential. Future work will experiment with completely self-controlled platforms for usability studies. If the expected user reaction is singular, only a temporal segmentation is needed. Regarding speech studies, the qualification of a word/pause detection module for the segmentation will be verified, to also label recorded utterances with the associated user action automatically.

The ideas presented may be supplemented by principles presented elsewhere [11]. In other works we introduced a language called *ICL* for rapid-prototyping of user interfaces. In their combination these concepts allow for fast creation and testing under realistic conditions and help in quick design of usable interfaces.

## V. Acknowledgments

The work presented in this paper has been supported by the FERMUS project, a cooperation of BMW Group, DaimlerChrysler, SiemensVDO and the institute of Human-Machine Communication at the TUM. The project stands for error-robust multimodal speech dialogues.

## VI. References

1. J. Nielson, „*Usability Engineering,*" Boston Academic Press, 1993
2. R. Sharma, C. Pavlovic, T. S. Huang, *"Toward Multimodal Human-Computer Interaction,"* Proc. of the IEEE, IEEE Vol.86, No. 5, pp. 853-869, 1998
3. J. Preece, *"Human-Computer Interaction,"* Addison-Wesley, 1994
4. S. Balbo, „*EMA: Automatic Analysis Mechanism for the Ergonomic Evaluation of user interfaces,*" Technical Report 96/44, 1996
5. D. Salber, J. Coutaz, *"Wizard of Oz Platform for the Study of Multimodal Systems,"* ACM INTERCHI'93 Conference on Human Factors in Computing Systems, Adjunct Proceedings p.95-96, 1993
6. G. McGlaun, F. Althoff, B. Schuller, M. Lang, *"A new technique for the individual normalization of workload in distraction tasks,"* CHI2002, Int. Conference on Human Factors in Computing Systems, Minneapolis, Minnesota, pp. 666
7. R. Nieschulz, B. Schuller, M. Geiger, R. Neuss, *"Aspects of Efficient Usability Engineering," IT+TI journal Vol. 44,* Oldenburg, pp. 23-30, 2002
8. F. Althoff, K. Geiss, G. McGlaun, B. Schuller, M. Lang, *"Experimental Evaluation of User Errors on the Skill Based Level in an Automotive Environment,"* CHI 2002, Int. Conference on Human Factors in Computing Systems, Minneapolis, Minnesota, pp. 782
9. F. Althoff, G. McGlaun, M. Lang, *"Combining Multiple Input Modalities for Virtual Reality Navigation - A user study,"* 9th Int. Conference on HCI, New Orleans, 20
10. F. Althoff, G. Mcglaun, M.Lang, *"Using multimodal interaction to navigate in arbitrary virtual VRML worlds,"* In WS on Perceptual User Interfaces (PUI 2001), Orlando, USA, Nov. 2001
11. B. Schuller, M. Lang, *"Integrative rapid-prototyping for multimodal user interfaces,"* USEWARE 2002, VDI/VDE #1678, Darmstadt, Germany, pp. 279-284