Towards a Development Methodology for Augmented Reality User Interfaces

Christian Kulas, Christian Sandor, Gudrun Klinker
Technische Universität München
Lehrstuhl für Angewandte Softwaretechnik
(kulas,sandor,klinker)@in.tum.de

ABSTRACT

In this paper we describe why we believe that the development of Augmented Reality user interfaces requires special attention and cannot be efficiently handled with neither existing tools nor traditional development processes. A new methodology comprising both a new process and better tools might be the best action to take.

A requirement analysis on issues regarding the process, the user groups involved, and the supportive tools for Augmented Reality user interface development is presented. This opens up a number of research challenges covering the tools, the process and the methodology as a whole. A new development process which is a first attempt to meet the newly found challenges is briefly outlined. This process relies on high parallelism and extends previously learned insights with usability evaluation matters. Following, our complementary proposed tool set gets introduced in detail. This set again profited mostly from new tools fitting in the usability engineering realm, which so far has been mostly ignored in the field of Augmented Reality. First steps towards a development methodology for the creation of Augmented Reality user interfaces, tackling the found requirements, are thereby made. Finally, our planed future steps are shown, meant to bring the development methodology further along, by solving important, but achievable, remaining challenges.

INTRODUCTION

One of the main activities in Augmented Reality user interface development, which is inherently multimodal, is the experimentation with different interaction techniques because it is such a young field. These have to be designed, implemented and evaluated. An important research issue here is to establish a development methodology that covers these three sub-activities and links them together more closely.

We believe the main groups of developers participating in Augmented Reality user interface development (Figure 1) are:

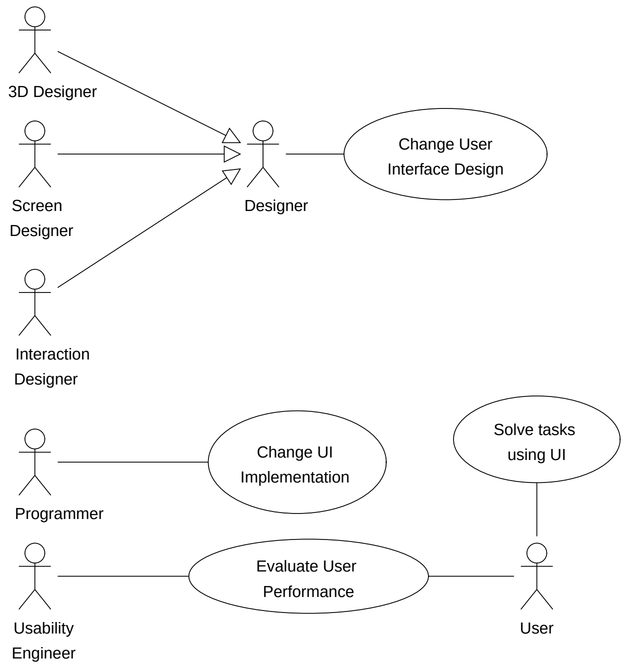


Figure 1: Use cases of all basic participating groups in the process (UML Use case diagram)

Programmer The programmer changes the implementation of the user interface by writing and editing low level code.

3D Designer This type of designer is concerned with creating 3D interaction / presentation elements which are to make an appearance in the user interface. An example for a 3D presentation element might be the landscape for SHEEP [11].

Screen Designer The focus of this designer is the actual screen layout, that is what is presented to the user at which location on screen.

Interaction Designer This designer wants to fine-tune the interactions the user can experience. She will set which multimodal inputs trigger which actions.

Usability Engineer The quality assurance of the usability of the user interface is point of interest for the usability engineer. This person combines all the roles of conducting usability studies in one. He selects, briefs, and debriefs the users for the study, prepares the evaluation materials, conducts and logs the actual study and finally analyzes and evaluates the results.

User The user actually uses the user interface by navigating through it in an attempt to accomplish certain tasks. For example she might want to place a roof on a building she is constructing within an architectural Augmented Reality application.

We identified several crucial requirements for each of these individual groups and for the development team as a whole that are presented in this paper. We see our work as first steps towards a methodology with a supporting set of tools for the development of Augmented Reality user interfaces addressing these requirements.

Building on our DWARF framework [2], we have already successfully tested [11] a new methodology for user interface design and implementation. The core idea was to let designers and programmers work simultaneously in the same room. In this paper we would like to present a new usability evaluation tool that allows us to further add simultaneous usability evaluation by usability engineers. The tool logs data about the user interactions and visualizes it to the usability engineers in real time, thus extending the work presented by Lyons and Starner [9].

The remainder of the paper is organized as follows: In the section *The Problem* we describe the requirements for a new development methodology. The section *Research Challenges* lists numerous open questions. The section *Our Approach* describes our prototypical methodology. Finally, the section *Future Work* discusses the next steps we intend to take.

THE PROBLEM

The development of Augmented Reality user interfaces requires special attention for multiple issues. These issues are presented in this section.

Process Issues

In traditional user interface development, a waterfall or an extended waterfall process (Figure 2) is followed [12]. Starting with the phase design, it is proceeded to the phase implementation and finally a phase evaluation. These phases are run highly sequential with no or little room for feedback or dependencies. This type of process works well if you have enough details about the design space and in general can anticipate implications of design changes well in advance.

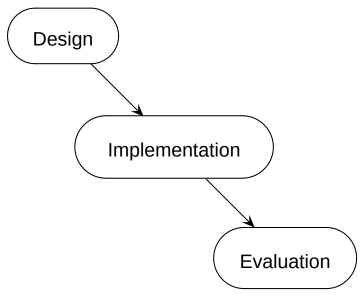


Figure 2: Basic waterfall process (UML Activity diagram)

Missing Tools

However, the task of developing Augmented Reality UIs is in itself still rather cumbersome due to the lack of tools to support the main three phases.

Design Authoring tools for design would accelerate the development significantly because they would allow quick assembling of user interface prototypes with various levels of functionality. Existing tools like Maya or 3DStudio can be leveraged by the 3D Designer, Adobe Photoshop or Microsoft Paint might be an aid for the Screen Designer. The support for the Interaction Designer is improving with projects like The Designers Augmented Reality Toolkit [10], which offers a collection of extensions to the Macromedia Director multimedia-programming environment, but still there is much work to be done.

Implementation The actual implementation of Augmented Reality user interfaces is made easier by a few frameworks such as DWARF [2], and Studierstube [15]. Implementation usually takes place in an Integrated Development Environment (IDE). Like mentioned earlier, first frameworks for designers are starting to emerge [10], but usability engineers are still kept in the dark.

Evaluation There are a number of imaginable tools, like automatic gathering and visualization of user performance data, allowing the quick generation of usability evaluation results, which could then be fed back to earlier phases. Unfortunately this class of tools is also in very short supply for Augmented Reality applications.

If we had proper tools like this, the generation of intermediate milestones would be much faster and thus make it more bearable to encounter problems in a post implementation evaluation phase, because a new iteration of design and implementation can be put together rather easily again.

Additionally, tools which actually do exist, usually only address their specific problem domain, with no or little integration with other related tools. In Augmented Reality applications, objects are registered in 3D [1]. Therefore, after completing a screen design, using a 2D tool, the designer usually needs to map the therein contained objects to 3D. She might have decided to keep a presentation component, keeping the user up-to-date on an important variable, like the amount of rescued sheep in a sheep herding application [11], in close reach, head-fixed [4] in the left right corner, all the time.

Currently, the screen designer has to re-create her earlier 2D design in 3D using a completely different tool for mapping 3D objects. It would be much more efficient if she could instead import her 2D design into a 3D registering application. But this is not possible without much better inter-tool integration.

Unclear Design Space

For traditional desktop software, vast amounts of knowledge on usability data exist, which created extensive and complete standard guidelines [16] which limit the design space to a known usable and working subset. Since Augmented Reality applications are a comparably new development, such a knowledge base is still to be built. So for now we are confronted with a vast design space and a big uncertainty which designs will work and which will not.

Unclear Non-Functional Requirements

Traditional software can also benefit from clear non-functional requirements. For example a web-site has to be navigateable, which meaning is defined in web-style guides together with all other non-functional requirements which are proven to be sufficient. But which non-functional requirements do we have to impose on Augmented Reality user interfaces? The graphical portions of the UI should probably be concise but what exactly does this mean?

Summary

Because of the lack of tools and uncertainties, traditional waterfall cycle processes are not suitable for developing Augmented Reality user interfaces efficiently. But even a new process cannot make up for the lack of suitable tools. So a new methodology based on both a better process and usable tools is needed.

RESEARCH CHALLENGES

The previously identified problems result in a number of research challenges on all areas tools, process and on the methodology as a whole which are the focus of this section.

On tools the main questions are:

- Which tools? There are numerous paths to take in supporting the main three user groups, resulting in a large design space for tools. It is a challenge to gain clarity regarding which type of tools will have the largest benefit.
- Tool integration? By integrating tools with each other, a much better work-flow between these tools can be leveraged building on tool chains. Where are the limits to integration and which integrations are reasonable at all?
- Tool mapping? Some tools might be useful to more than one user group thanks to a high level of integration. The presentation of multiple tools simultaneously to certain user groups might have more value, than the sum of each single tool on it's own merit. It is a challenge to figure out which tool combinations map best to which user groups.
- Tool automation? The more knowledge on UI design is accumulated, the more ideas for automation features in tools can be generated. For example, basic clear cut design principles which have been shown to apply in certain scenarios could be enforced in design tools. Since we still lack knowledge in this area, it is unclear which automations will be indeed feasible in the future. Instead of testing against known usability problems, there have been interesting approaches in web interface development, like WebRemUSINE [13], which try to automatically identify new usability problems. This is done by looking at the level of correspondence between how users actually perform tasks and the intended system task model. This idea might also be applicable to Augmented Reality user interfaces.

On the process the main challenges are:

- Limit to parallelism? By conducting multiple different development phases at the same time much better feedback can be attained. But how parallel can the process get without losing reasonability? The different phases of the process have undeniably certain dependencies which will probably not allow a total parallel execution.
- Formal process? Only by obeying a formal process similar to Extreme Programming [3], built on reasonable rules and process patterns [5], a highly parallel development can be accomplished. But which practices do apply the best on Augmented Reality user interface development?
- Persistence of UI experiments? It is in the nature of rapid proto-typing to experiment with different variations of the UI in quick succession. However, after testing a number of UIs, it is very desirable to be able to roll-back into a previously evaluated UI iteration since it may have turned out to be best suited after all. It is a challenge to build the process in a way to ensure the results of these prior experiments are not lost.

Finally, on the methodology as a whole:

- Limits? Is the new methodology only suited to create prototypes for temporary experimentation or might it actually yield usable products which can be deployed at the site of the customer?
- Validation? Does the methodology actually fully meet all requirements we impose on it? Answering this is also a challenge, since the exact definition of requirements is still a non trivial task.

Going deeper into the research challenges, we will now take a closer look at the tool questions. The issues regarding the process and the methodology as a whole cannot be considered in any more detail until more future work has been done.

Tool Combination Design Space

In an attempt to tackle some of the tool research challenges, it is helpful to correlate a list of likely supporting tools with all three groups in a matrix like found in table 1. Following this, the value of the different pairs can be assessed to learn which challenges are the most worthwhile to be confronted first.

IDE or Authoring

A 2D Paint Tool is probably only beneficial to the screen designer such as a 3D Modeller also probably cannot serve anyone but the 3D designer. Basically these are already authoring tools. An authoring tool for interaction could of course also help the interaction designer putting together new interactions. Generally, any integrated development environment or authoring tool should be a great benefit to all three user groups if they are adopted enough to their requirements. For any programmer an Integrated Development Environment (IDE) is already a standard tool to rely

Tool	Designer	Programmer	Usability Engineer
2D Paint Tool	+	-	-
3D Modeller	+	-	-
IDE or Authoring	+	+	+
Performance Logging & Visualization	-	o	+
Wizard of Oz	+	+	+
Automatic Testing	+	+	+
Monitoring Tool	-	+	+
Interaction Graph	+	+	+

Table 1. Tool combination design space matrix

on. Likewise, the usability engineer could use an authoring tool to put together user performance visualizations or to define tasks for the user to attempt which performance is then automatically measured.

Performance Logging & Visualization
User performance data is only directly interesting to the usability engineer and only has an indirect impact on the designer and programmer. However, the programmer might benefit from this feature, too. If it was automated enough to do some initial rough tuning concerning usability, the programmer might be able to later skip implementation code changes fixing obvious usability flaws.

Wizard of Oz
A Wizard of Oz [14] tool could actually benefit all three user groups. The interaction designer could use it to test in advance if certain interactions pan out or not, before actually prototyping them. The programmer could leverage this tool for feature dummy implementations to make early integration tests between partially incomplete features. The usability engineer could conduct simulated full-featured usability studies by faking yet missing functionality to gain insights into usability aspects, which would normally not be attainable until much later into the implementation phase.

Automatic Testing
Automatic testing tools of various sorts would again benefit all three user groups. There could be a tool to check for conformity of standard design guidelines, which would ease the life of the designers by avoiding trivial design errors. A similar tool to JUnit could quickly double check mandatory functionality, after code refactoring has taken place by the programmer. Once enough usability knowledge in the area of Augmented Reality has been accumulated, hard usability guidelines might crystallize themselves. Their conformity could be again tested against by automatic tools, which would remove the strain of the usability engineer to evaluate these then trivial usability parameters. This enables him to focus on the still unclear and less studied usability questions.

Monitoring Tool
A monitoring tool visualizing the state of the distributed Augmented Reality application and the communication between all components should obviously be useful for a programmer. The usability engineer could also profit from such a tool, if it was integrated with performance visualization. He would use the tool to indicate which data he is currently most interested in, which is then visualized.

Interaction Graph
In multimodal interaction, inputs from different media channels trigger defined actions. For example, by combining a speech token with a gesture, a wall could be deleted in an architectural Augmented Reality application. A tool could visualize this interaction graph, display received tokens and in general show the progress of the user in his current task. Such a tool could be beneficial to all three user groups. The interaction designer could use such a tool to visualize his work and even create new interaction paths with it. The programmer could use it to learn at which interaction his code got stuck to ease debugging. Finally the usability engineer could use such a tool to learn in which interaction the user is currently struggling, if this is not obvious through other means.

Summary
As a result, we are confronted by a large amount of research challenges covering multiple areas, of which we only had chance to look closer at tool questions for now. Even here it is still unclear if the list of tools is complete and how much the tool integration can cover. By evaluating tool with user group correlations, initial ideas were gained which missing tools seem to be the most promising. Although our focus is mostly on tools, we will now briefly cover our process approach in the next section after which our attempt at meeting a number of tool needs is detailed, too.

OUR APPROACH
To make up for the lack of tools, a better process offering much more feedback between phases is necessary. In fact we believe that only maximizing this feedback can offer us enough efficiency until our knowledge base is large enough to allow older, slower paced, sequential processes.

To maximize this feedback, we propose to run all three phases design, implementation and evaluation in parallel (Figure 3).

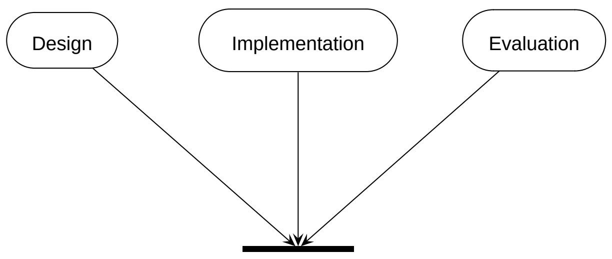


Figure 3: Proposed parallel process (UML Activity diagram)

We already learned a few valuable lessons regarding the process within the earlier SHEEP project [11]. In Jam sessions, development takes place at run time obeying same time & place principles. This allows a *crowded group* working with peer code reviews and on-the-fly insertions of altered system components, for quick integration tests. These sessions proved to increase the development speed significantly. This process also allows playful exploration, because sub-components which have an impact on the user experience can be swapped during run-time effortlessly, enabling quick assessment of different variations. Iterative, continuous development is an implication of this.

When a high level of tool integration is achieved to support the efforts of all user groups in all three development phases a very fast, feedback-driven and parallel process to develop Augmented Reality user interfaces like proposed might be indeed realizable.

Now, our newly developed supportive tools for the usability engineer are presented in detail, after which a few other older tools are also briefly introduced. Finally, a possible tool combination for the usability engineer is presented.

Usability Evaluation Tools

At Technische Universität München we have developed a framework for usability evaluations in the field of Augmented Reality, covering both process as well as software issues, applicable on applications based on DWARF [8].

Our momentary focus lies on the therein newly developed software tools, but before presenting these in detail it is worthwhile to give a quick overview of the intended process for conducting usability evaluations by looking at a possible room setup (Figure 4).

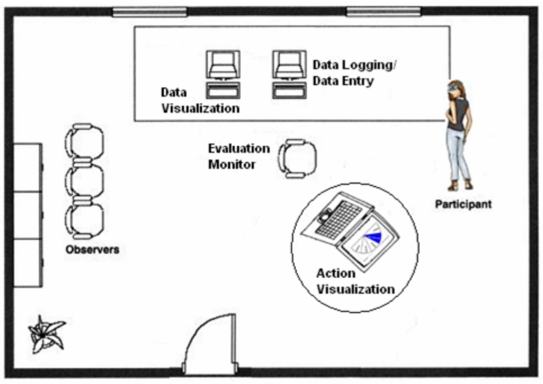


Figure 4. Room setup for usability evaluations

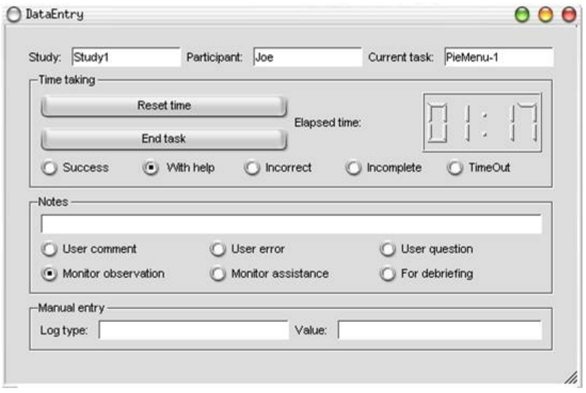
The setup might look like this at *crowded group* working when even an end user is taken into account while debugging the system. The user is placed at a suitable distance of the usability engineer / evaluation monitor who is busy entering observations (Data Entry) into the usability logging system (Data Logging) while also monitoring what the user actually sees on screen (Action Visualization) and while monitoring real-time visualizations of measured usability data (Data Visualization).

Multiple peers or the evaluation monitor himself might at the same time observe internal system behavior and even fine tune the system on the fly while observing usability implications immediately.

This lab-based approach is usually considered as “Local Evaluation” with both the user and the usability engineer in the same place at the same time. We believe, this is still the best way to capture qualitative usability data on Augmented Reality user interfaces. However, when Augmented Reality systems are used on a more frequent basis globally, a remote evaluation approach using Remote Usability Evaluation Tools [7] might be more reasonable. Here the system usually presents a wizard-based dialog to the user, asking her details about her opinion on the usability problem after recognizing a critical usability incident [6] automatically or after the user triggered the dialog herself. By design, this requires quite a lot of effort on the part of the user herself. Additionally, great care has to be taken regarding issues of user privacy when passing on collected data without her prior consent.

With this in mind, the mentioned software components are now covered in more detail. The core component is a fully automated logging tool to capture events from the running Augmented Reality system. For this it is important to mention that Augmented Reality systems, which are built modularly leveraging the DWARF framework, communicate internally mostly by means of CORBA based events running through event channels which can be effortlessly tapped into by any logging tool interested in doing so, such as the newly developed logger.

A manual data entry tool (Figure 5) can be leveraged to take quick written notes for later review, which are also directly passed on to the data logging component.



DataEntry

Study: Study1 Participant: Joe Current task: PieMenu-1

Time taking
Reset time
End task
Elapsed time: 01:17
Success [✓ With help] Incorrect Incomplete TimeOut

Notes
User comment User error User question
[✓ Monitor observation] Monitor assistance For debriefing

Manual entry
Log type: Value:

Figure 5. Dialog to enter usability data manually

All performance measurements can finally be visualized in real-time during usage with a number of highly flexible and adaptable scripts.

Figure 6 shows a number of different sample visualizations. It was decided to base the visualization off the GNU General

Public licensed (GPL) third party tool *ploticus* [1] for multiple reasons. Its' scripting language proved to be well suited for rapid prototyping of new visualizations while maintaining a high level of ease of use. Additionally, it already had all the 2D plotting support we required, that is it supports out of the box all standard 2D plotting styles including line plots, filled line plots, range sweeps, pie graphs, vertical and horizontal bar graphs, time lines, bar proportions, scatter plots in 1D or 2D, heat-maps, range bars, error bars, and vectors.

Numerics, alphanumeric categories, dates and times (in a variety of notations) can be plotted directly. There are capabilities for curve fitting, computing linear regression, and Pearson correlation coefficient r. There is a built-in facility for computing frequency distributions. Means, medians, quartiles, standard deviations, etc. can also be computed out of the box meeting our needs for default statistical functions.

For the first sample study we conducted [8], the four shown visualization types have been assembled.

Before elaborating the details of these different types, the usability data log file format must be exposed. It is a standard ASCII file in which each line represents exactly one log file entry. Each entry must have six components to gain unambiguous data. The first mandatory component encompasses the detailed current *date & time* of the log entry for later time dependent data mining. The second component stores the *study* name, since multiple studies are to be conducted, which are not to be mixed up. For similar reasons and to facilitate intra-user comparisons and task time taking, *user* and *task* names are stored in the third and fourth component. The most interesting components are the two last ones since they store the logged event *type* which might be e.g. a Button-click and its' corresponding *value* e.g. Hit or Miss.

Leveraging this log file format, the visualization types from top left to bottom right shown in Figure 6 are:

Relative Error
This script is the least flexible, since it requires the *value* fields to be exactly Hit or Miss for the to be analyzed *study*, *user*, *task*, and *type* combination. It visualizes the resulting accumulated hit-ratio (y-axis) over time (x-axis). The final hit-ratio is additionally printed separately in a box.

Task Time Range
Requiring only the specification of the to be analyzed *study*, a range of all task (x-axis) completion times (y-axis) averaged over all participating users is visualized. These times are extracted from the log file by filtering for special event *types*, which mark the begin and end of any given *task*.

The actual ranges can be easily visualized in different ways. Shown is the mean and standard deviation. The biggest dot indicates the mean times while the error bars extend to the standard deviation. The smaller light dots show the individual task completion times of all *users*. The stars denote task completion times outside of the standard deviation. Finally below each *task* a number is printed, which depicts the number of averaged tasks, which is equal to the amount of *users* who took this *task*.

[1] `http://ploticus.sourceforge.net`

We also prepared a median version which renders a big dot at the median time for each task while the box-plot extends to the 25th and 75th percentile. Error-tails extend to the border values while smaller light dots show the individual task completion times for all *users*.

All range visualization parameters can be easily adopted on a case-by-case basis.

Value Timeline
This visualization has the same parameter requirements like the *Relative Error* visualization. Here it is merely shown which event type *value* (y-axis) occurred at what time (x-axis).

Figure 6 actually shows a slight modification of this basic visualization. An additional line visualizing a study specific additional event *type* and *value* development was added effortlessly to be able to better spot usability flaws of a specific nature [8].

Absolute Bars
Requiring the *study*, *task* and *type* parameters, absolute totals (y-axis) of all different *values* (x-axis) are rendered in horizontal bars. If no *user* is specified, it will output average bars together with a specification on how many users the script averaged over. However, if a *user* name is given, it will output bars using data from this specific user only.

Sample usability study results [8] using the above tools are out of scope for this paper but it should be mentioned that our sample study was very promising.

Other Tools

In this section other older tools developed by us, which support the first two phases are briefly introduced because they will offer insights on future integration.

Our framework for multimodal interactions is an UI architecture described by a graph of multiple in-/output and control components (User Interface Controller (UIC)) ([11], Figure 7 bottom-left).

The UIC can be visualized, and since it shows the complete user interface interaction graph, it is very useful for interaction designers. This tool is a very close approximation of the interaction graph tool, mentioned earlier. However, it is still nowhere feature complete.

The arbitrary event streams within DWARF, as well as all participating communicating components can be visualized by our general-purpose monitoring tool DWARF Interactive Visualization Environment ([11], Figure 7 bottom-right) which currently serves as a debugging tool for programmers.

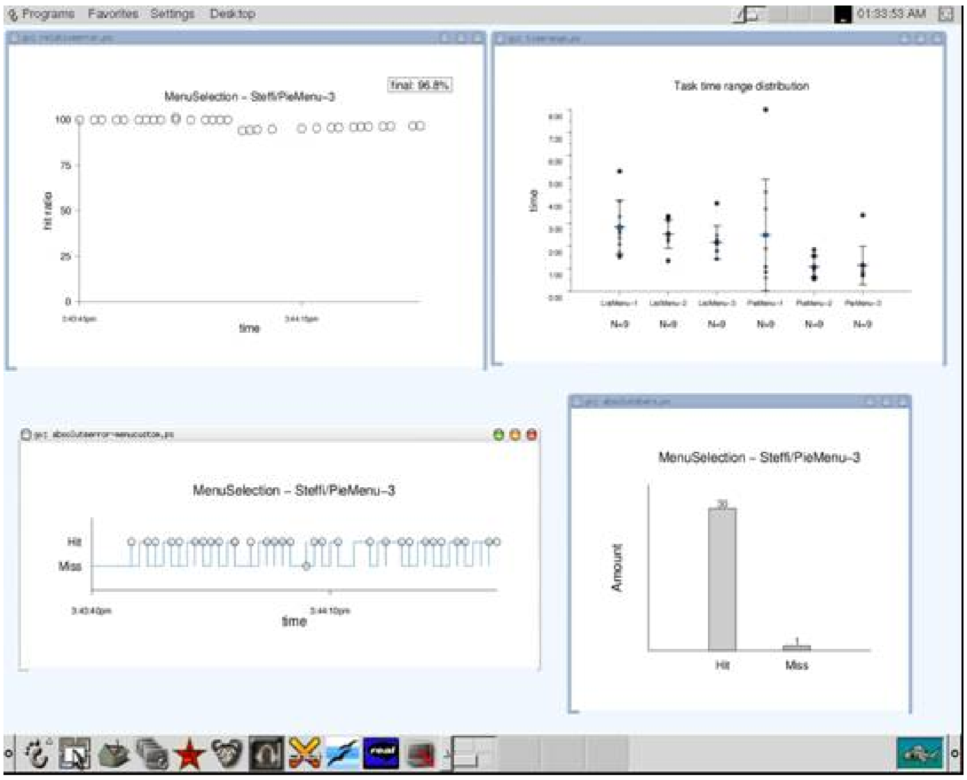

Programs
Favorites
Settings
Desktop
01:33:53 AM
MenuSelection - Steffi/PieMenu-3
final: 96.8%
hit ratio
time
Task time range distribution
time
N=9
MenuSelection - Steffi/PieMenu-3
Hit
Miss
time
MenuSelection - Steffi/PieMenu-3
Amount
Hit
Miss

Figure 6. Sample real time visualizations of logged usability data

Given all these tools, a combination which should hopefully prove to be useful to the usability engineer in future scenarios is now presented.

Usability Engineer Tool Combination

In Figure 7 the UIC, the monitoring tool and the user performance real-time visualizations are combined on one screen.

The monitoring tool (bottom right) shows raw unfiltered event communication between service components while at the same time showing all running services with full details on their states. The current version of our monitoring tool is really only useful for the programmer, but extensions are imaginable which make this worthwhile for the usability engineer after all (see section *Future Work*). The UIC (bottom left) might help the usability engineer to understand where the user is currently within the interaction graph.

Finally the real-time user performance measurement windows (top row) should enable the evaluation of the actual usability at the same time. While the first two tools could although reveal that a certain action was successfully triggered by the user, it does not become apparent how many tries there were, at which time frame, or how many errors there have been until this final tool is taken into consideration.

Observations in the top windows will likely usually lead to implementation fine tuning to e.g. trigger actions differently or they might reveal the need for a whole new service to e.g. install a data filter for better usability, thereby in effect overhauling the design.

FUTURE WORK

There are still many challenges to solve providing us with multiple objectives for future work. Of course it is future work to implement the missing tools and achieve a high level of integration to be able to better follow the proposed methodology. One of the first easiest integrations to do is to add Wizard of Oz functionality to our UIC.

Additionally, the monitoring tool could be integrated with the user performance visualizations to make this tool feasible to the usability engineer. Currently, a fixed set of scripts for visualization have to be pre-selected prior to the study by the usability engineer which will then be constantly updated with live data. However, it would be much preferable if the usability engineer could change these visualizations on-the-fly by e.g. clicking on a map representing the system state and exchanged events, similar to what the monitoring tool already offers to DWARF.

An authoring tool for the interaction designer is another very critical next step, since this type of user still has clearly the worst tool support. Being overly visionary, this tool could even reach bootstrapping proportions. That is the designer starts out with a very basic authoring tool based on Augmented Reality and uses this tool to extend itself by building new widgets which can create ever so bigger interactions slowly creating a full-blown user interface.

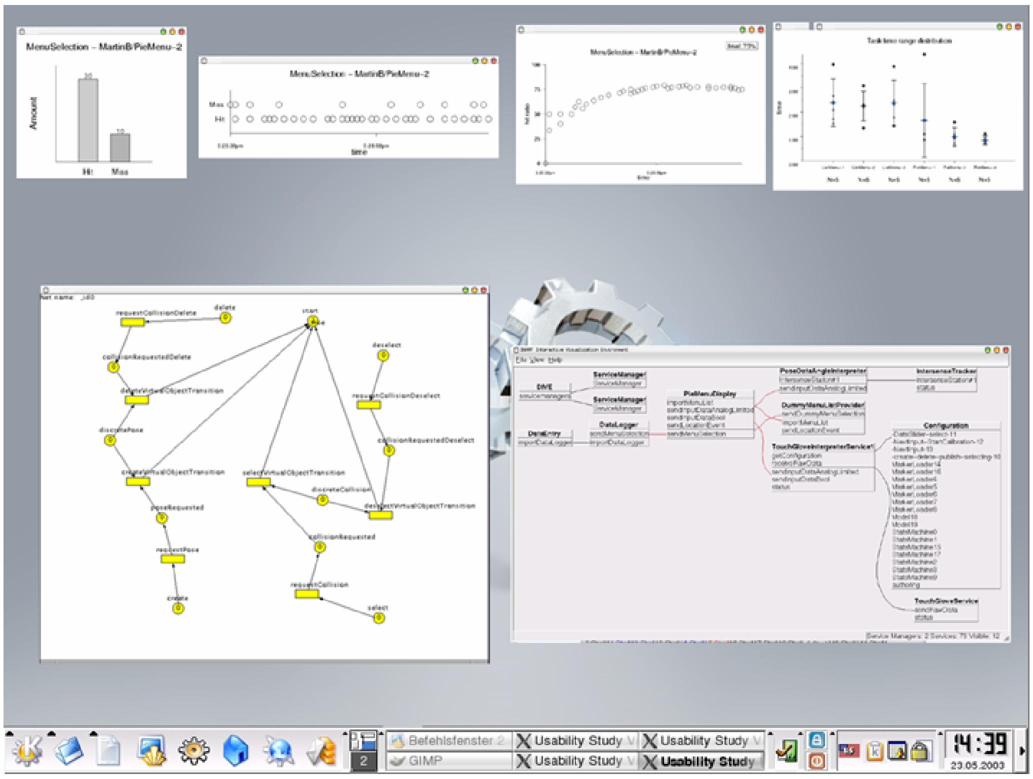


Figure 7. Usability engineer tool setup mockup

Currently we only aim at mastering a better process of manually designing, implementing and evaluating user interfaces for Augmented Reality applications, but in the future we will also want to invest in proactive UIs. Here the application evaluates itself during runtime and changes its' own user interface design, and corresponding implementation, automatically on-the-fly. For example, by observing user behavior patterns over time, it would be possible to take note of never used functionality which could be hidden to generate a less obstructing UI. The process itself needs much more refinement which will be achieved by conducting more experiments gradually accumulating hopefully in a formal model.

In summary, traditional development processes and current tools are ill-suited for Augmented Reality, and only by improving on both the process as well as developing new or integrating existing tools a more suitable platform for creating Augmented Reality user interfaces can be established.

REFERENCES

1. R. T. Azuma, *A Survey of Augmented Reality*, Presence, 6 (1997), pp. 355–385.
2. M. Bauer, B. Bruegge, G. Klinker, A. MacWilliams, T. Reicher, S. Riss, C. Sandor, and M. Wagner, *Design of a Component–Based Augmented Reality Framework*, in Proceedings of the 2nd International Symposium on Augmented Reality (ISAR 2001), New York, USA.
3. K. Beck, *eXtreme Programming Explained: Embrace Change*, Addison-Wesley, 1999.
4. S. Feiner, B. MacIntyre, M. Haupt, and E. Solomon, *Windows on the World: 2D Windows for 3D Augmented Reality*, in ACM Symposium on User Interface Software and Technology, pp. 145–155.
5. A. Granlund and D. Laf, *A pattern-supported approach to the user interf ace design process*, 1999.
6. H. Harston and J. Castillo, *Critical Incident Data and Their Importance in Remote Usability Evaluation*, in Human Factors and Ergonomics Society 44th Annual Meeting, pp. 590–593.
7. N. Kodiyalam, *Remote Usability Evaluation Tool*, Master's thesis, Virginia Polytechnic Institute and State University, 2003.
8. C. Kulas, *Usability Engineering for Ubiquitous Computing*, Master's thesis, Technische Universität München, 2003.
9. K. Lyons and T. Starner, *Mobile Capture for Wearable Computer Usability Testing*, in Proceedings of IEEE International Symposium on Wearable Computing (ISWC), October 08-09 2001, Zurich, Switzerland, pp. 69–76.
10. B. MacIntyre, J. D. Bolter, E. Moreno, and B. Hannigan, *Augmented Reality as a New Media Experience*, in Proceedings of the 2nd International Symposium on Augmented Reality (ISAR 2001), New York, USA.
11. A. MacWilliams, C. Sandor, M. Wagner, M. Bauer, G. Klinker, and B. Brügge, *Herding Sheep: Live System Development for Distributed Augmented Reality*, in Proceedings of ISMAR 2003.
12. D. J. Mayhew, *The Usability Engineering Lifecycle*, Morgan Kaufmann Publishers, 1991.
13. L. Paganelli and F. Paternò, *Intelligent Analysis of User Interactions with Web Applications*, in ACM Symposium on Intelligent User Interfaces, San Francisco, CA, 2002.
14. T. Reicher and T. Kosch, *Software Design Issues for Experimentation in Ubiquitous Computing*, in The Second Workshop on Artificial Intelligence in Mobile Systems (AIMS 2001), Seattle, Washington, USA, August 4, 2001.
15. D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavari, L. M. Encarnação, M. Gervautz, and W. Purgathofer, *The Studierstube Augmented Reality Project*, Presence, 11 (2002).
16. B. Shneiderman, *Designing the User Interface*, Addison-Wesley Publishing, 1997.