Cause and Effect in User Interface Development

Ebba Thóra Hvannberg
University of Iceland
Hjardarhaga 2-6
+354 525 4702
ebba@hi.is

ABSTRACT

There is a lack of means of translating or relating work products from elicitation, such as work models, to design and using results of evaluation as feedback to design. This paper suggests that a richer model of evaluation be created that is built concurrently with the design activity and that records the cause / effect relationship between design and the problem domain and the implications work models have on design. It also suggests that the distinction between elicitation and evaluation be diminished. The paper presents two case studies from air traffic control and poses questions that are meant to motivate researchers and practioners.

Categories and Subject Descriptors

H5.2 [**User Interfaces**]: *prototyping, evaluation/methodology, theory and methods.*

General Terms

Design, Experimentation, Human Factors

Keywords

Prototype, Development Lifecycle, Air Traffic Control, Model, Change

1. INTRODUCTION

Life cycles of user centred user interface development are well known and consist of eliciting user needs and their environment, specifying the user and organizational requirements, producing design solutions followed by evaluation, usually in several iterative cycles in an interdisciplinary team [8]. The four basic activities have been researched and practiced by developers often with good results.

How information flows between these four activities is not as well known and we hypothesize that this is the reason for the lack of interplay between evaluation and design. As in any activity, the four activities have input and output. The input is the basis for the activity and the output is the deliverable of the activity and usually input into the next activity in the lifecycle. The output of elicitation can be user, task or goal models (work models) of various types, and description of actors and their environment, i.e. context. The output of the design activity is one or several design ideas for a feature realized in low to high fidelity prototypes, a model or a final system. The output of the evaluation activity can be failures detected, hindrances, facilitators, and positive or negative consequences of a designed feature. The lack of means of translating output, coming either from elicitation or evaluation, to design ideas is an obstacle in the lifecycle of development of user interface.

If a design for a feature is rejected, it can be difficult to decide how it should be changed. Then we need to go back to the drawing board to create new design ideas. In software development, finding root causes has been widely used and the CUP (Classification of Usability Problems) [7] method has been suggested to further classify attributes of failures in user interaction and to find their roots in processes of the user interface development lifecycle. To find causes of problems (e.g. undesirable effects), i.e. backwards at the time of evaluation, we may record the cause and the desired effect at the time of design. The causes may be miscellaneous and even multiple; they can be within the design features or the underlying work model. Hence, one should also note the implications a work model is meant to have on design. (see Figure 1). In this paper, we set forth research questions that have emerged from our work in prototyping and evaluation of two case studies in air traffic control. The aim of presenting the case studies is to examine the activities and learn how they can be a basis for discussion of a development lifecycle and in particular its work products. The next section gives an overview of two design experiments where low-fidelity prototypes have been used. Examples in the remainder of the paper are taken from the case studies.

NordiCHI'04, October 24th, 2004, Tampere, Finland.

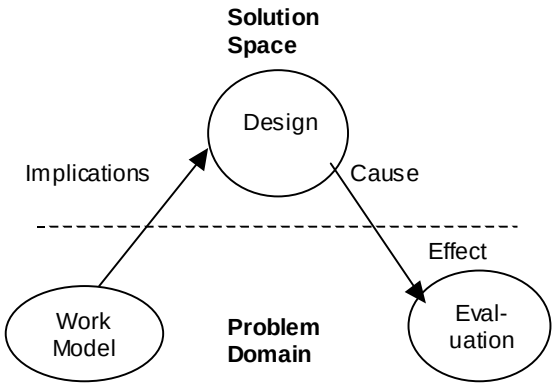

Figure 1 Cause in Design and Effect in Evaluation

Table 1 Methods used

	Speech Agent	**Integrated workstation**
Elicitation	Literature review Observation Interview	Observation Interview Existing systems & requirements studies Class & Collaboration diagrams Cognitive models of user's work Heuristics evaluation using cognitive principles
Design	Architecture Sequence diagrams Prototype	Paper sketches Three alternative approaches suggested
Evaluation	Wizard of Oz with air traffic controllers Post-test questionnaire Qualitative and Quantitative data gathered	Claims analysis Walk-through of drawings of user interface with participation of air traffic controllers post-task questionnaire Qualitative data gathered

2. CASE STUDIES

The two case studies reported here are taken from the domain of air traffic control. The duty of the air traffic controllers in the studies is to service aircraft en route in oceanic environments, i.e. cross the North-Atlantic. They monitor aircraft against predetermined routes, but issue clearances for requests for different routes provided it is safe, i.e. if aircraft adhere to separation rules. In the following two subsections, we describe how elicitation, design and evaluation were carried out in the two projects. Table 1 provides an overview of the methods used.

2.1 Using language technology to improve communication in ATC

2.1.1 Elicitation

Previous literature on voice communication in Air Traffic Control was analysed [10]. Oceanic Air Traffic Controllers were observed while at work at a centre of air traffic control and operators at a Centre of Radio Communication were observed. The researcher interviewed expert controllers to learn about the domain of Air Traffic Control and to understand the role of voice communication. The challenge in this domain is that fortunately errors in voice communication are relatively infrequent so they are not easily observed.

2.1.2 Design

A prototype of a speech agent was developed with the goal of recognizing errors in the communication between pilot and controller. Several options to replace or add a speech agent to existing voice communications were explored and their architecture designed, but a prototype of only one was implemented.

Sequence diagrams describing realistic scenarios, edited by expert users, of dialogues were created for three characteristic scenarios of the problem domain.

2.1.3 Evaluation

A Wizard of Oz evaluation was conducted with five controllers of varying expertise. The evaluation was scripted, using the dialogues described in the previous section, with the tester playing the role of the pilot against each of the controllers. Quantitative data was gathered on errors made by the speech server during evaluation and quantitative and qualitative data on controllers' attitude towards trust and performance was gathered in a post-test questionnaire. The evaluator asked questions about the type of feedback a speech agent should give in case of error in the voice communication between controller and pilot.

Since the prototype was of low fidelity, it was not feasible to evaluate it in context, other than to create real life scenarios and to have actual users. Evaluators were not conducted in Air Traffic Controllers' room or in a group with collaborators of the work, such as controllers of the same centre, supervisors, or controllers of adjacent centres. Although this is considered important, and perhaps especially so when researching voice communication, it would have been impossible to get permission for evaluation on site and hence would have to be staged.

2.1.4 Results

Since the tests had to be scheduled in advanced, and resources were scarce, there was no time to pilot test the evaluation on site. Hence, some of the evaluation instances were flawed because of failures in the supporting technology. The script worked very well, the performance of the speech agent was measured and the controllers were able to understand and reflect on the concepts. Controllers' attitude towards expected efficiency, safety and their trust on the speech agent has to be viewed in context of the artefact evaluated, but were acceptable to proceed to the next phase. Performance of the speech agent gave the designer good ideas on how to improve its design and implementation.

2.2 Integrating different user interfaces in a controller's workstation

2.2.1 Elicitation

As in the previous case study, observations were made, but a wider range of controllers was interviewed [9]. The architecture of different subsystems of a workstation was analysed including their relationships.

An abstract model of the problem domain was created based on manuals of operations, previous requirements studies, observation of work and current systems. The model was expressed with text and UML diagrams.

A user interface model was reengineered from two current systems in order to find possible anomalies and basis for integration of two user interfaces.

Cognitive models of user's work were examined.

Heuristic evaluation, using cognitive principles, was carried out on current ATC's workstation to find deficiencies.

2.2.2 Design

Three alternative approaches to integration were described but one of them designed in detail as drawings of user interfaces.

Snapshots of user interfaces of design ideas for several features were created in a drawing tool. Snapshots were ordered into a short storyboard explaining a scenario of work.

Except for the description of the integration of the three alternatives, no models of designs were made, neither as scenarios, interactions, navigations, dialogues nor structure of user interfaces. The reason may was that the focus was on limited design features illustrated at the presentation level.

2.2.3 Evaluation

Evaluation did not take place in context, except that interviewees were air traffic controllers. Controllers were asked to give a preference to one of three alternative approaches to integration of the two user interfaces. A researcher conducted claims analysis [11] of three alternative approaches to integration.

Evaluations of snapshots were made with controllers of varying expertise. No interaction took place but instead the researcher described situations to users. For some features several alternatives were presented and users asked to rate them and discuss, but for others only one design was presented. The method of evaluation was an interview with predetermined questions about safety, performance, and invited design suggestions from the controllers. Two iterations of evaluations took place with feedback from the former affecting the latter.

2.2.4 Results

The snapshots of designs of user interfaces provided valuable means for interviewing users about the new ideas. Researchers received good ideas from users and the two iterations showed that improvements were achieved. The triangulation of evaluation methods, i.e. claims analysis and users' preference gave researchers additional confidence in the results.

The abstract models drawn and the cognitive models examined during elicitation were both useful to understand the complex problem domain and to explore new design ideas for specific aspects. They were particularly helpful in moving away from current context, which was necessary because the technological and consequently other contextual layers are changing.

3. ELICITING NEEDS AND CONTEXT

In this and the following two sections, we describe the activities of the user interface development lifecycle. We end each section with questions or challenges that will help us link the activities. Prior to the questions, we give examples from the two case studies. The first activity in a human-centred design is to understand and specify the context of use. Contextual inquiries [3] and ethnographic approaches have been gaining popularity in recent years. Less is known about how to produce work products that are useful for software engineers or user interface designers. Context, partnership, interpretation, and focus are four principles that guide contextual inquiry. The first and most basic requirement of Contextual Inquiry is to go to the customer's workplace and observe the work. The second is that the analysts and the customer together in a partnership understand this work. The third is to interpret work by deriving facts, make hypothesis that can have implication for design. The fourth principle is that the interviewer defines a point of view while studying work. The output of this activity can be e.g. a work model and Beyer and Holtzblatt [3] suggest several models that comprise the work model, i.e. a model of communication, a sequence model, an artefact, or cultural and physical models. The lack of formalism in these models makes them difficult for practioners like engineers to adopt. Semi-formal models in UML could replace or complement these informal models.

Vicente [12] argues that work analysis for systems should identify and model intrinsic work constraints, and that the models should have formative implications for design. The motivation is that there is no systematic way to go from results of testing to prototype attributes and therefore we are dependent on the creativity of the designer to revise the prototype to remove the problematic effect. The CWA (Cognitive Work Analysis) is an example of such a formative approach to work analysis and so is the Contextual Design proposed by Beyer and Holtzblatt [12]. Above we listed the models of Contextual designs that are created, but CWA presents other conceptual distinctions [12, p. 120]: *Work Domain*, *Control Tasks*, *Strategies*, *Social-Organizational* and *Worker Competencies*. Through analysis of these distinctions, models of intrinsic work constraints are created that again lead to system design interventions. We give examples of interventions for Strategies, Social-Organization and Worker Competencies. Dialogue

modes and process flow are based on constraints derived from strategies. Role allocation and organizational structure are based on Social-Organizational constraints. Training and interface form are based on constraints derived from worker competencies. Neither Vicente nor Beyer and Holtzblatt express explicitly or maintain in a formal way the design implications of the work analysis. Vicente gives informal relationships in between the two activities by taking examples, but work analysis and not design is the subject of [12]. More often than not, motivations for system implementation are changes. Those changes are e.g. due to changing technological contexts of the problem domain, increased scale, increased demand for quality or changing technological changes in the solution space. Below, is an example that shows how proposed changes in Social-Organizational conceptual distinction has an implication to a design.

A simplified example from the speech agent
Social-Organizational: A speech agent replaces a radio operator.

How can the implications of work analysis to design be modelled and maintained?

4. DESIGN

The data collected during elicitation and evaluation of previous versions of the modified problem context will guide new design ideas. Design can be abstract such as re-design of work or structure of information, to detailed interactions between a product and the context. The design of the user interface is of this last type.

Before a user interface is programmed, we can create a model of the design that we use to evaluate against our requirement and assumptions. The model may range from being abstract, like diagrams or wire frame, or detailed, such as sketches. Prototypes of various types, i.e. low vs. high fidelity, experience prototypes [5], vertical and horizontal, throwaway and incremental prototypes, are popular since they give the user an idea about the look and the feel of the interface. Other products can be used to model certain aspects of a user interface such as navigation, dialogue or architecture such as diagrammatic models e.g. in UML or extensions thereof. Storyboards and textual scenarios are often useful to present design ideas or concepts respectively early on.

Designers should select the type of model that is most appropriate for the design feature at hand. For example, when designing complex navigations, a navigational diagram that gives an overview of the traversals between contexts will be more useful than many detailed sketches of designs. On the other hand, when designing presentations for entities that contain a rich collection of information, sketches are more useful. A complex dialogue implementing a scenario may be best presented with both sketches and diagrammatic models.

Speech agent: The Wizard of Oz prototype was supported by a sequence diagram describing scenarios that were evaluated.

How can we guide designers to use a combination of different design products, such as different fidelities of prototypes, diagrammatic models, text scenarios, or text use cases?

4.1 Multiple Design Ideas

One of the fundamental principles of design is to create multiple design ideas for a feature. This can be a result of a brainstorming session with an interdisciplinary team including users. When the design team has been a participant in the whole lifecycle, design ideas are implicitly linked to user needs and context of work.

The rationale for the design idea needs to be made explicit. Otherwise it will be difficult during evaluation to validate weather the design feature is coherent with the problem domain. Evaluation of the design is prepared during the design phase. In our experience, it is not adequate to ask whether a design meets requirements of efficiency, effectiveness and satisfaction especially for design ideas produced early in the life cycle, often in low fidelity prototypes. Designers should associate evaluation questions with the design ideas during design but not after it. Thus, usability specialists should either work on the design team, or in small organizations designers, should take on the role of usability test designers.

Integration of ATC: Three alternative design ideas for integration were presented. Claims analysis was applied to elicit positive and negative consequences. Questionnaires were posted to elicit views on usability of the alternatives.

How can we design and describe evaluations of user interfaces that can answer specific questions about the effect of the design?

4.2 Tradeoffs

Design ideas are created to change a problem domain. There may be different motivation for the change, i.e. technical, social, organizational, or economical. Common effect of the changes that we are aiming for are increased effectiveness, efficiency or satisfaction during operation. Other changes may result in increased safety or less time for training. A design idea that may cause a positive effect of one aspect of the problem domain may at the same time cause a negative effect of another. We take an example from combining two user interfaces, Flight Data Processing (Flight strips) and Radar Data into one. The merging of the two interfaces will eliminate the need to integrate information in the user's head but can increase clutter on the display. More automation in the Flight Controller workstation can lead to less workload in easy low traffic situations but may blur the controller's mental picture (leading to less efficient or safe operations) during difficult high traffic or critical situations. The above statements are similar to claims analysis [11] where positive and negative consequences of a single design feature are gathered. Bass and John [1] describe how we can analyse tradeoffs of different software architecture patterns and their effect on usability.

Integration of ATC: Controls for selecting altitude levels cause the controller to focus on specific critical air traffic and reduces the cognitive load, thereby making decisions easier.

How can we express the expected effect in the problem domain, resulting from changes brought on by the design ideas?

Integration of ATC: Controls for selecting altitude levels cause the controller to miss information in deselected altitudes and therefore deteriorating the mental model of the current state of the system.

As we see above, when creating different design ideas, there can be tradeoffs between them. Another example is taken again from ATC. Either adaptable (i.e. adapted by the user) or adaptive (adapted by the computer) user interfaces are meant to solve the problem of display clutter that can occur during high traffic situations.

Integration of ATC: An adaptive interface can be more efficient than adaptable interface to the controller but less satisfying.

How can we express tradeoffs of effects between design ideas?

5. EVALUATION IN CONTEXT

The goal of the evaluation is to see how the proposed changes interact with the problem domain. Hence, by introducing the new design, we have modified the problem domain.

Many methods of evaluation have been proposed, both analytical and empirical, most are manual but some are also automatic. The results are either qualitative but also quantitative. Evaluations are done at different phases in the development life cycle, but close interaction with users from an early stage has been advocated. Evaluating finished products may be easier but failures detected at such a late stage may be costly to correct. Hence, designers have focused on early evaluation with low-fidelity prototypes, experience prototypes with users. The down side is that these evaluations may not be as reliable e.g. in safety critical situations.

Although contextual inquiries have been advocated, there has been less emphasis on evaluations of design in real contexts and, in the case of early evaluation, this may prove to be unfeasible. However, experience prototyping [5] has been proposed as a tool to use for this purpose. Every effort should be made to place the design in real contexts. Training facilities can be used to accomplish this. Simulators may be another way.

Speech agent: Controllers were recruited to participate in the evaluation, scenarios were carefully designed and verified to emulate real contexts.

How can we build a context for evaluation of designs during early phases?

The results of an evaluation can be twofold; either the design ideas were not able to correctly fulfil the assumptions or the model of needs or the underlying model of the problem domain proved incorrect. This results in changing the model or changing the design. In the former case we might have assumed something about the work or its context, but found out during subsequent evaluation that the assumption was not correct. An example from the speech agent is that we assumed that controllers spoke at a specific speed with no delays. This assumptions lead to a certain configuration in the agent. This is an example of a relationship between how some knowledge about he problem domain leads to a design decision. During evaluation, it became evident that the assumption was not correct. If we have a model of the relationship between the problem domain and the new design ideas, it will be easier to trace back the causes of failures, correct the underlying model, and adapt the design. Not all such relationships may be evident beforehand and some are only realized during evaluation.

Although we may have specified the expected effect of a design idea, it may be that it will lead to some actual unforeseen effect. The evaluation in context is about finding out how the new design ideas interact in the changed problem domain. Hence we try to observe what changes the ideas bring about to the entire problem domain, not only the immediate user, but other systems and stakeholders.

How can we express the actual effect in the problem domain, resulting from changes brought on by the design ideas?

If we fail to reach the desired effect, we may either trace backward to the documented causes of the desired effect or else we need to trace it to failed designs or wrong assumptions in the problem domain.

How can failures (in reaching the desired effect) lead us to failed designs (causes) or wrong assumptions in the work model?

6. DISCUSSION

This paper has presented challenges that need to be addressed to better integration evaluation and design. The approach proposed involves specifying different work products. We have used two case studies to illustrate our challenges with simple examples, expressed above in boxes. They are by no means meant to be examples of how to address these challenges, but rather give some initial illustration of the concepts.

Although it is easier for developers to understand the lifecycle consisting of separate activities and we understand that it is important to have several iterations of the activities, the gap between them may be unnecessary.

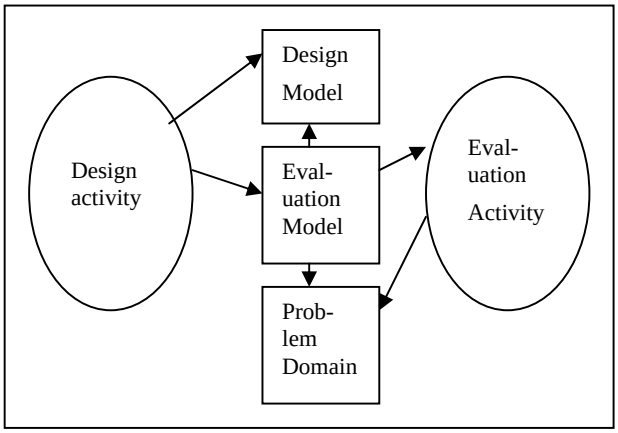

Figure 2 Development model

We propose (see Figure 2) to have two activities and that the Design and Evaluation activities are run concurrently, with the two artefacts Design (and/or a model thereof), the Model of the Problem Domain, and The Evaluation Model as central repositories. The distinction between elicitation and evaluation may not always be clear since evaluation elicits new information and gives us further data about user needs and their

environment. The only difference between them is that at elicitation usually (but not always) no design of features is presented. This constitutes the first iteration, but in subsequent iterations, we use the term evaluation because some product of the design has entered the domain.

The Evaluation activity should not be conducted as a separate activity after the Design, but instead planned for during Design and then carried out. We have a practice in software development where it is recommended to design the test before the implementation. Extreme Programming [2], which is a type of an agile development methodology, has this practice as one of its main guidelines. Cockburn [6] offers two advantages of automated regression tests: the developers can change the code and retest it very easily and there is less stress if the developers can run automated regression tests since they are then ensured that no one else has altered the code. Unfortunately, it is difficult to write such automated test for user interfaces, and hence the more reason to attempt to make them formal and easily repeatable. Briand et al. [4] have proposed a revision of the Goal Quality Metric framework, called GQM/MEDEA that adds empirical hypotheses and aims to make them quantitatively verifiable.

7. ACKNOWLEDGMENTS

Margrét Dóra Ragnarsdóttir and Hlynur Jóhannsson have designed and evaluated the prototypes of the speech agent and integration of user interfaces respectively.

8. REFERENCES

[1] Bass, L., John, B. Linking Usability to Software Architecture Patterns through General Scenarios, *The Journal of Systems and Software, 66 (2003) 187-197*

[2] Beck, K., Test-driven development, Addison-Wesley, 2002

[3] Beyer, Hugh and Holtzblatt, Karen, *Contextual Design*, Morgan Kaufman, 1998

[4] Briand, L. C., Morasca, S., Basili, V. An Operational Process for Goal-Driven Definition of Measures, *IEEE Transactions on Software Engineering, vol. 28, no. 12, December 2002*

[5] Buchenau, M., Suri J. F., Experience prototyping, DIS'00, ACM, 2000

[6] Cockburn, A., *Agile Software Development,* Addison-Wesley, 2002

[7] Hvannberg, E.T, Law, L. C., Classification of Usability Problems (CUP) Scheme, Interact'03, IFIP, Switzerland, 2003

[8] ISO/IEC 13407:1999 Human-centred design processes for interactive systems

[9] Johannsson, H., Hvannberg, E.T., Integration of Air Traffic Control User Interfaces, 23rd DASC, Digital Avionics Systems Conference, IEEE, 2004

[10] Ragnarsdottir, M. D., Waage, H., Hvannberg, E.T., Language Technology in Air Traffic Control, 22nd DASC, Digital Avionics Systems Conference, IEEE, 2004

[11] Rosson, M.B. and Carroll, J. Usability Engineering: Scenario-Based Development of Human Computer Interaction, Morgan Kaufmann, 2002

[12] Vicente, Kim J. *Cognitive Work Analysis*, Lawrence Erlbaum associates, 1999