Time-Based Decision Trees in Interaction Design

Diploma Thesis at the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

In Cooperation with the
Department of Psychiatry
and Psychotherapy
University Hospital Aachen

by
Sascha Beckers

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Dr. PD Ute Habel

Registration date: Dec 11th, 2007
Submission date: July 28th, 2008

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed, and that I have marked any citations accordingly.

Hiermit versichere ich, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, July 28th, 2008

Contents

List of Figures

List of Listings

Abstract

This thesis describes the design of the *Presentation Visual Editor*, a new user interface for the stimulus delivery and experimental control software system *Presentation*. The Visual Editor is a graphical development environment for the creation of psychological experiments. It allows the visual creation of experiment structures and direct layout of visual stimuli.

The design is based on time-based decision trees which are used to model time structures in interactions. Time-based decision trees are graph representations of hierarchical decision rule systems that include time conditions.

Before starting the development of the system, I conducted a Contextual Inquiry to get an understanding of the psychological domain and to learn about the users' working methods and problems in the operation of Presentation. The users were observed in the field while implementing experiments.

The design was evolved in three cycles of an iterative user-centered design process. In the first iteration, a paper prototype was designed to test the users' understanding of the concept and to collect fundamental feedback about the interactions of the interface. This paper prototype was then refined and evaluated again in the next iteration. Finally, I have implemented a working interactive prototype in Java to test the interface in more detail.

A final evaluation compared the Presentation Visual Editor with Presentation's original interface and measured the performance of the design. This study showed that time-based decision trees are beneficial for the design of psychological experiments. The thesis closes with a discussion of the significance of the results for general interaction design.

Überblick

Diese Diplomarbeit beschreibt die Entwicklung des *Presentation Visual Editor*, einer neuen Benutzeroberfläche für das Softwaresystem *Presentation*, das der Stimulusüberbringung und Experimentsteuerung dient. Der Visual Editor ist eine graphische Entwicklungsumgebung zur Erstellung psychologischer Experimente. Er erlaubt die visuelle Erstellung von Experimentstrukturen und die direkte Gestaltung von visuellen Stimuli.

Das Design basiert auf zeitbasierten Entscheidungsbäumen, die zur Modellierung von Zeitstrukturen in Interaktionen benutzt werden. Zeitbasierte Entscheidungsbäume sind Graphdarstellungen von hierarchischen Entscheidungsregelsystemen, die Zeitbedingungen beinhalten.

Bevor ich mit der Entwicklung des Systems begann, führte ich eine kontextabhängige Untersuchung durch um ein Verständnis für die psychologische Domäne zu entwickeln und um die Arbeitsmethoden und Probleme der Benutzer bei der Bedienung von Presentation in Erfahrung zu bringen. Die Benutzer wurden hierzu während der Implementierung von Experimenten im Feld beobachtet.

Das Design wurde in drei Zyklen eines iterativen, benutzerzentrierten Designprozesses entwickelt. In der ersten Iteration wurde ein Papierprototyp entworfen um das Verständnis der Benutzer für das Konzept zu testen und um grundsätzliches Feedback über die Interaktionen in der Benutzeroberfläche zu sammeln. Dieser Papierprototyp wurde danach in der nächsten Iteration verfeinert und erneut evaluiert. Schließlich habe ich einen funktionierenden interaktiven Prototypen in Java implementiert um die Benutzeroberfläche genauer zu testen.

Eine Endevaluierung verglich den Presentation Visual Editor mit Presentations ursprünglicher Benutzeroberfläche und maß die Leistung des Designs. Diese Studie zeigte, dass zeitbasierte Entscheidungsbäume sich vorteilhaft auf das Design von psychologischen Experimenten auswirken. Die Diplomarbeit schließt mit einer Diskussion über die Signifikanz der Ergebnisse für allgemeines Interaktionsdesign.

Acknowledgements

A lot of people contributed to this diploma thesis and I am very grateful for that. This project would not have been possible without their help and support.

First of all, I would like to thank Prof. Dr. Jan Borchers for the opportunity to write this thesis at his chair and my supervisor Jonathan Diehl for his constant support and feedback and the many ideas he has contributed to my work.

Special thanks also go to Dr. Ute Habel for being my second examiner and for always showing interest for my work and giving feedback. I am also very grateful to Nils Kohn and Timur Toygar, my contact persons in the Department of Psychiatry and Psychotherapy, for introducing me into the world of psychological research and for their great support and help and the many ideas they had for my work.

Thanks to everybody at the Media Computing Group. I really appreciated the good and friendly working atmosphere. Especially, I want to thank Eileen Falke for reading my thesis and Noriyasu Vontin for his feedback on several issues.

Further, I would like to thank everybody at the Department of Psychiatry and Psychotherapy for allowing me to do this project with them and for warmly welcoming me. Special thanks go to all participants of my various evaluations and user studies for giving valuable feedback and spending their time for my project.

I also want to thank Eva Temur for reviewing the entire document.

Last but not least, I want to thank my wife Anna for all her emotional support and patience during the time of this thesis and during my entire studies. You gave me the strength to do this, thank you!

Conventions

Throughout this thesis I use the following conventions.

The whole thesis is written in American English.

Unidentified third persons, for example users or participants of evaluation studies, are always described in male form. This is only done for purposes of readability.

Chapter 1

Introduction

"Time is what we want most, but what we use worst."

—William Penn

Time aspects are an important element in the design of interactive systems. The designers of user interfaces (UI) of interactive systems should not only consider elements, such as the visual design or the accessibility of functions. To achieve good usability and therewith successful systems, it is important to consider the temporal layout of the interaction. This user interface time controls the whole course of the interaction. It determines when which actions are taken or not taken and how the system responses to the behavior of the user. If the temporal layout is designed properly and different kinds of time constraints are applied, it is possible to create sophisticated system behavior. Time constraints could treat, for example, continuous inactivity or repeated errors of the user.

It is important to design the temporal layout of interactions

Therefore, it is crucial to have means for modeling the UI time in the design process of an interactive system. The modeling of time should be efficient and easy understandable for humans. Although there are many mature software development environments for the creation of interactive systems, the efficient modeling of time is an unsolved problem which does not seem to receive serious consideration.

Means for modeling UI time needed

Current development tools neglect UI time

The focus of today's development environments is primarily on the spatial layout of UIs, i. e. on the arrangement of interface elements and the assignment of actions to them. Software modeling techniques, like Unified Modeling Language (UML) diagrams, model time only for the visualization of internal processes and system architecture. Thus, these techniques all ignore UI time constraints. The lack of appropriate design tools causes software developers to disregard the explicit consideration of time issues and their effects on the interaction flow during the design process.

Time-based decision trees proposed as solution

In this work, I will explore a new design metaphor based on time-based decision trees. It can be used to model time in interactions. Standard decision trees are graph representations of hierarchical systems of decision rules. A time-based decision tree extends this notion by including time into the decision process. Decisions made by the tree depend additionally on time constraints and the tree must obey to time flow. To test the performance of this new concept, it will be realized in a highly specialized integrated development environment (IDE).

I will develop the IDE, called *Presentation Visual Editor*, in cooperation with the Department of Psychiatry and Psychotherapy at the University Hospital Aachen. This clinic conducts psychological and neuropsychological experiments for their research. They implement these experiments with the application *Presentation*.

Presentation is an experimental design and control software system

Presentation is a stimulus delivery and experimental control software system for neuroscience used for programming, performing, and analyzing psychological experiments. It is a common system and very powerful, but it bares some serious disadvantages. The Presentation UI, consisting of two scripting languages, frustrates its users and even harms their research because the difficulties they experience influence and constrain the design of their experiments. The underlying languages, the Scenario Description Language (SDL) and the Presentation Control Language (PCL), needed for specifying the course of events, are responsible for laborious experiment realization. SDL and PCL base on traditional software techniques, whereas time aspects are of great importance in psychological experiments. Altogether, in the design of the program the demands of the domain were not taken into account. This makes Presentation an excellent application for taking advantage of time-based decision trees.

Research question

The main goal of this work is to show that the use of time-based decision trees is beneficial in interaction design. This will be achieved by creating design metaphors for using temporal decisions and applying them in the design of an improved user interface to Presentation.

The Presentation Visual Editor user interface, which I will present in this work, has the following properties:

Development tool for interactive systems

- It is a development tool for creating a very specialized form of interactive systems: software-controlled psychological experiments.

Bases on Presentation

- The interface bases on Presentation and has the goal to output executable SDL/PCL code at the end.

Time-based decision trees model experiment structure

- In its design time-based decision trees are incorporated to explicitly model time constraints and the time flow of the experiment. The trees serve as means for making the whole time structure visible and less abstract.

GUI system

- In contrast to Presentation, the Presentation Visual Editor has a real graphical user interface (GUI). The complete experiment construction process will be performed with visual interactions and without programming. Optional programming of custom experiment features would be possible in the generated code, though.

Visual creation of stimuli

- The entire process of stimulus creation will be facilitated with visual interactions that allow direct manipulation.

Evaluated in comparative study

- The performance of the design will be evaluated in a comparative study that contrasts the Presentation Visual Editor prototype directly with Presentation's original interface. This will show whether the time-based decision tree concept has benefits for the design process.

User-centered design approach

My design approach will be guided by the principles of iterative user-centered design. This should guarantee good usability and an intuitive design as the users directly participate in the design process. The design will evolve in several steps. The end product of this work will be an interactive high-fidelity prototype of the improved Presentation interface that can be used to create experiments with visual stimuli.

1.1 Chapter Overview

Chapter 1 introduces to the topic of this work. It explains why the modeling of time in the design process of interactive systems is important and an unsolved problem. Further, it describes the proposed solution to this problem and introduces the Presentation Visual Editor user interface that will be designed to test the approach. This chapter explains the context and the motivation of this specialized interface and lists its fundamental characteristics.

Chapter 2 covers the theoretical background for this work. It starts with an explanation of the main concepts of (neuro-) psychological experiments. After introducing general decision trees and time-based decision trees, I will explain the concept of timed automata and argue why decision trees are preferred in this work. The next section covers the importance of time constraints for humans and interactive systems. Moreover it illustrates the current state in software development. Then I will describe the important concept of iterative user-centered design. The chapter concludes with an examination of the software requirements of domain experts.

Chapter 3 presents example systems that incorporate decision trees—with and without the consideration of time conditions. However, there is no system that is directly related to my concept. Furthermore, I will describe SMIL, a language for creating Internet presentations, and SuperLab, a graphical experiment design system, which is an alternative application to Presentation. I will describe its characteristics and argue why the Presentation Visual Editor is superior.

Chapter 4 investigates the current state of experiment design with Presentation. After explaining the system in detail, it deals with the Contextual Inquiry I conducted to learn about the users' problems with Presentation. I describe the method and the design of the study and finally present the findings that had great influence on the first prototype.

Chapter 5 is about the first design cycle in the development process. It presents the design of the first prototype of the Presentation Visual Editor, which is a paper prototype, and its evaluation. The prototype aimed at fundamental high-level feedback. My goal has been to learn whether the users understand the time-based decision tree concept and whether the interactions are intuitive. The feedback provided many suggestions for improvement and valuable qualitative results about the features of the interface.

Chapter 6 describes the second paper prototype of the interface. Motivated by the evaluation results from the previous chapter, I made several modifications to refine the design. In addition, I incorporated some new design ideas. The aim was to construct a clearer and simpler interface before implementing a working system. The focus of the evaluation was on gathering feedback about the design changes.

Chapter 7 presents the final prototype of the Presentation Visual Editor, an interactive software prototype implemented in Java. The design of the system was again refined considering the previous evaluation results. The prototype was evaluated in a pilot study that aimed at obtaining more detailed feedback. Its purpose was to prepare the final evaluation. I had to ensure that the interactions are understood by the users and that no unexpected problems occur.

Chapter 8 describes the final evaluation of the interface. I conducted a comparative study that contrasted the Presentation Visual Editor prototype with Presentation's original interface. In addition, the users were supposed to fill in a questionnaire. The study allowed me to measure the performance of the design and to collect quantitative and qualitative feedback.

Chapter 9 concludes this work. It summarizes the results of the project and gives an overview of the issues that should be treated in the future. I discuss how several issues can be implemented in future development to solve problems of the interface and to facilitate experiment construction. Further, this chapter addresses open problems that were out of the scope of this project. Finally, it is discussed to what extent the results of this work can be generalized to interaction design.

Chapter 2

Theory

"It doesn't matter how beautiful your theory is, it doesn't matter how smart you are. If it doesn't agree with experiment, it's wrong."

—Richard P. Feynman

This chapter presents the theoretical background that is needed to understand my work. First, I will give an introduction to psychological experiments. Then I will explain decision trees and their extension to time-based decision trees and will distinguish them from timed automata. Afterwards, this chapter covers the importance of time constraints for humans and interactive systems. Finally, I will explain iterative user-centered design, an important concept this work adheres to, and discuss the special requirements of domain-expert users for software design.

2.1 Psychological Experiments

The interface, which I will develop in this work, has the function to design and to implement psychological experiments. Therefore, this section will briefly introduce the main concepts of designing and performing psychological experiments. This will give first insights into the psychological domain. The description of the experiments is mainly based on the book of Huber [2005].

Psychology deals with human behavior, experience, and consciousness

Zimbardo and Gerrig [1999] define the subject of psychology as the behavior, experience, and consciousness of human beings and their development over the span of life and their interior and exterior conditions and causes. The goals of psychology as science are the description, explanation, and prediction of behavior. Since psychology is an empirical science, controlled experiments are of utmost importance to verify the validity of theories. Popper [1998] developed seminal theories for the empirical research process. He says that the correctness of scientific theories can never be verified through scientific testing. He takes falsifiability as the criterion of demarcation between what is and is not genuinely scientific. A theory is scientific if and only if it is falsifiable. The fundamental concept of empirical research is thus the critical testing of theories with regard to falsification. This implies that a well-tested theory must be accepted as true until it is falsified.

Falsification basis for empirical research

2.1.1 Hypothesis Test and Variables

Hypotheses predict experiment results

The primary goal of psychological experiments is to test hypotheses for their truth content. A hypothesis is an assumed answer to a solvable scientific problem that can be tested. In particular, a hypothesis must be falsifiable. Hypotheses are propositions about correlations and causalities of phenomena.

Three types of variables: IV, DV, EV

There are three types of variables in experiments: independent variable (IV), dependent variable (DV), and extraneous variable (EV). The IV is a variable that is varied actively by the experimenter. IVs are deliberately manipulated to invoke a change in the DVs. The DVs are the variables that are observed to change in response to the IVs. The DV's reaction is the event that is predicted in a hypothesis. Finally, EVs are variables that may probably also influence the DV. Their effect must be neutralized, because they would interfere with the IV's effect. This leads to the following definition of an experiment: In an experiment the experimenter varies systematically at least one IV and observes the effect of this variation on at least one DV. At the same time, he eliminates the effect of EVs.

EVs must be neutralized

An example for a hypothesis is: Learning with breaks is more effective than learning without breaks. In this case the IV is the existence of a break and the DV is the learning success in a subsequent test. EVs could be, for example, fatigue, previous knowledge, and motivation of the subjects.

Keep EVs constant or vary randomly

There are several methods for controlling EVs. The two main strategies are keeping the values of the variables constant for all subjects in the whole experiment and in contrast not keeping one level but combining randomly several different levels of the EV with the values of the IV. To control possible EVs of the subjects, the variables can be measured for each subject. Then the subjects are divided in different groups so that the average values of the groups are as similar as possible (matching). Another strategy is to divide the sample randomly into different groups. In order to control variables of the experimental situation, one can try to eliminate the variable completely, e. g. installing sound insulation to eliminate distracting noises. Here again, the EVs can be kept constant or can be randomized. Another strategy to control influences of EVs, which mainly stem from the selected group, is the inclusion of a control group, which carries the EVs in a normally distributed extent.

Parallelized or randomized groups

Eliminate variables

Control group

Multiple experimental conditions cause EVs

It should be mentioned, if the subjects are exposed to multiple experimental conditions, additional EVs emerge. They result from the absolute or relative order of conditions. Those EVs can be, for example, learning effects, increasing fatigue or hunger in the course of an experiment that influence the results of later experimental conditions.

2.1.2 Operationalization and Measuring

Operationalization makes concepts observable

A hypothesis contains (theoretical) concepts. Concepts cannot be directly observed. But the hypothesis should be tested empirically. Therefore, the concepts have to be operationalized. Operationalization denotes the assignment of observable and measurable phenomena to concepts. The operationalizability of hypotheses is a requirement for their testability. Often there are multiple possibilities for operationalizing concepts. Therefore, the researcher has to pay attention to find a operationalization of good quality that really measures the desired variable. For further discussion see Amelang and Zielinski [1994].

Several methods of data acquisition

Observation

There are several operationalization techniques, also called methods of data acquisition. The basic method is scientific observation. Ultimately, only through observation data can be acquired. In modern scientific observation complex devices, such as computed tomography (CT), are often employed. Behavioral monitoring is a form of scientific observation that is very meaningful for psychology. For example, the behavior during interactions between students and teachers can be observed.

Survey

The next technique is the method of survey. Here the subject answers to questions. Surveys can be more or less structured. It can be a free interview or given answer categories that have to be checked. Surveys can be conducted verbally as interviews or written as questionnaires.

Test

Finally, data can be acquired with tests. In a test, subjects are given standardized stimuli, such as test pictures, test exercises, or test questions, under standardized conditions. The different operationalization techniques can, of course, be combined in an empirical research. Special psychological operationalization techniques can be used as well as techniques from other disciplines in one experiment. For instance, in neuropsychology the researchers employ neurophysiological procedures to operationalize brain activity.

Measuring assigns numbers to objects

Variables can have many different values. A means to express the intensity of variables is measuring. A requirement for measuring variables is that they are operationalized. Measuring assigns numbers to measured objects in such a way, that specific empirical relations between the measured objects are represented as specific numerical relations between numbers. There are four issues in measuring. The representation problem is about whether an empirical variable is measurable. The uniqueness problem is about how the measurements can be transformed (e. g. in a different unit); dependent on the level of measurement specific transformations are allowed or not. In psychology four levels of measurement are used: Nominal allows the determination of equality. Ordinal allows ordering of values. Interval allows statements about the ratio of intervals be-

Four levels of measurement: Nominal, ordinal, interval, ratio tween measurements. Ratio allows in addition statements about the ratio of measurement values. The third issue in measuring is about which conclusions can be drawn on the basis of measuring, i. e. which statements about the values are meaningful. This also depends on the level of measurement. Finally, when doing measurement the issue arises how to scale the values concretely.

2.1.3 Experimental Design

Logical structure of experiments

The experimental design is the logical structure of an empirical test with regard to the hypothesis test. The experiment has to be planned in such a way that the intended hypothesis test becomes possible. The design is very important, in particular when the effect of multiple IVs is studied simultaneously.

Subjects can do one or multiple experiment variants

Basically, there are two ways of designing an experiment. When choosing the between-groups method, each subject only does one variant of the experiment. When choosing the within-groups method, the subjects are exposed to multiple or all experimental conditions. In the easiest case, one IV with two possible values is tested. Here, one experimental design is to work with two groups of subjects, one for each experimental condition. After the experiment, the results of the two groups can be compared. The group that is treated with the experimental condition that is of interest to the researcher, is called treatment group. The control group allows the comparison and controls EVs. Analogously, when testing one IV with more than two values, more groups have to be established. In another possible design, only one group of subjects is used that is exposed to two or more conditions. Because of learning and other position effects, this design is not generally acceptable for experiments.

Often the phenomena that should be studied in an experiment are more complex. This requires testing the effects of multiple IVs at once. In an experiment, in which multiple IVs are varied simultaneously, the different values of the different IVs are combined with each other. For example, if there are two IVs with two and three values respectively, there are six possible combinations. Thus, there are six experimental conditions which require six treatment groups. Of course, here again it is possible, that subjects do multiple variants. This has the advantages, that less subjects are needed and that EVs of the individual subjects are neutralized—unfortunately this is not always possible and multiple conditions can cause additional EVs.

Experiments in the field vs. experiments in the lab

When designing an experiment, it has to be decided whether the examination takes place in the lab or in the field. Experiments in the field, i. e. in the subject's natural environment, create more realistic situations and behavior may be more natural. Therefore, the results can be easier utilized for the application in specific natural situations. But the control of EVs and the operationalization of IV and DV is much harder in the field.

Experiments in the lab, i. e. in a special examination room, have the advantages that the experimenter has better control over EVs and over the operationalization of IV and DV. On the contrary, it may be harder to generalize the results on natural situations. In general, most experiments are conducted in a lab, because only there the experimenter has access to special equipment that is needed.

2.1.4 Samples

Samples should be representative

The researcher has to choose a sample of subjects after operationalizing the hypothesis, constructing an experimental design, and controlling the EVs. A sample is a subset of the population being examined and is chosen for participating in the experiment. It should be as representative for the population as possible.

Samples can be chosen randomly or deliberately

A sample can be chosen with random processes or deliberately. In practice, it is often a compromise between these two methods as it is too expensive to choose an actual random sample of the population in which every person has the same chances to be chosen or simply impossible due to confounding effects when searching for participants or due to the experimental design. Researchers often take a very limited random sample, e. g. consisting of students of the own university. But the better the sample, the better can the results be generalized, because the random process ideally leads to normally distributed EVs. A further method is choosing a stratified sample that reflects the distribution of the population with respect to a certain variables, such as age or profession. Difficulties with stratified samples are that many variables are hard to operationalize and that data about the distribution in the population is required. Stratification and randomization can be combined when searching subjects, but as mentioned before, in practice samples often are neither really stratified nor randomized.

Samples can be stratified

2.1.5 Empirical Test

Formulation of empirical prediction

In order to test a hypothesis empirically, the researcher has to formulate an empirical prediction based on the hypothesis. This prediction has to be compared with the results of the test. But the EVs can still interfere with the IV. Thus, the systematic effect of the IV has to be distinguished from unsystematic effects of the random EVs. This can be achieved by the application of statistical theories. Consequently, the researcher must derive a statistical hypothesis from the empirical prediction. This statistical hypothesis will finally be tested in the experiment. When analyzing the data obtained in the empirical test, an appropriate statistical evaluation procedure has to be performed which will result in accepting or rejecting the statistical hypothesis. Afterwards the researcher can draw a conclusion about the hypothesis.

Derivation and testing of a statistical hypothesis

Reliability, objectivity, validity

Thereby it is important to consider quality factors of the experiment: The results must be reliable, i. e. different researchers will always measure the same; the results must be objective, i. e. independent of the concrete experiment procedure; and the experiment must be valid, i. e. it must be ensured that it was really tested what was meant to be tested. Additionally, the quality of the operationalization, the quality of the control of EVs, and the quality of the chosen sample are important factors when interpreting the results of a psychological experiment [Amelang and Zielinski, 1994].

2.1.6 Neuropsychological Experiments

Neuropsychology explores psychological functions in the brain

The department of psychiatry and psychotherapy of RWTH Aachen University mainly conducts neuropsychological research. Neuropsychology is a branch of psychology that explores how the structure and function of the brain relate to specific psychological processes [Kolb and Whishaw, 2008]. Most commonly used measures are neuroimaging techniques. Nevertheless, there are also paper and pencil tests or psychophysiological measures which relate to and describe brain functions and therefore are used in neuropsychological research.

fMRI measures neural activity

An important neuroimaging technique is functional magnetic resonance imaging (fMRI) . fMRI captures variations in magnetic fields. It measures the hemodynamic response supposedly related to neural activity in the brain. When neurons become active, they consume oxygen carried by hemoglobin. The local response is an increase in blood flow and blood volume, occurring after a delay of a few seconds. The hemodynamic response, which is analyzed in many cases, rises to a peak over 4-5 seconds, before falling back to baseline. These changes have effects on the magnetic field, which can be measured with fMRI. The method for detecting magnetic field variations due to differences in the oxygen saturation of the blood is called blood-oxygen-level dependency (BOLD). The BOLD signal indicates which brain areas are active at a given time. When subjects repeatedly perform certain thought processes or actions, statistical methods can be used to determine the areas of the brain which reliably are more active during that process. fMRI has a high spatial resolution in the range of millimeters. Due to the inertness of the hemodynamic response, the temporal resolution is, however, only in the range of seconds [Schneider and Fink, 2007, Müsseler and Prinz, 2002].

Example study: Neural correlates of emotion

In the following I will give an example for a possible fMRI research, that is motivated by an actual study. The example study is about neural correlates of emotion. Therefore, an experiment about mood induction is designed. This can be done with presenting happy faces, texts, music, or positive odors to the subjects for specific intervals. In the experiment, subjects are treated with different experimental conditions. Besides a condition in which happy faces are shown to induce positive emotions, in one condition a neutral stimulus, such as a "+"-sign, could be presented in order to cause as little neural activation as possible (a low-level baseline). Another treatment could consist of presenting "scrambled" faces that do not have a recognizable face and thus, do not convey any emotion. In this way, there is a high-level baseline since seeing faces causes neural activity. All the brain areas are activated that are responsible for processing the perception of faces, but the brain areas that are correlated to emotion are not activated. The contrast between these measurements and the measurements taken while showing emotional faces indicates which brain areas are correlated to emotion.

Block design and event related design

For fMRI research, there are basically two kinds of experimental design: block design and event related design. These are two different approaches to obtain the activation contrasts in fMRI measurements. In block design the brain activation is usually contrasted during a time interval longer than six seconds (the duration of one hemodynamic response) with another interval that either is meaningful, i. e. there are two tasks that differ in the element of interest, or that causes little activation (low-level baseline). In event related design only one hemodynamic response function with a defined onset is observed after presenting a stimulus. Thus, only the activation during one neural response is contrasted with non-defined scans which are seen as baseline. The response function can be shifted, stretched, or compressed, but it suffices to scan for a maximum of about seven seconds, which is shorter than in block design.

Block design has the advantage, that several scans are cumulated to obtain stronger activation measures, e. g. for a mood induction effect. Thus, it is suitable for measuring effects that are supposed to last for some time or whose time of occurrence cannot be anticipated. The measuring over longer time, however, captures also events that are not of interest.

In event related designs, the onset of a function is clearly determined, e. g. by a given stimulus. The activation can be measured almost immediately. Event related design has the disadvantage that many individual events are needed to achieve significant results (see Horwitz et al. [2000]).

2.2 Decision Trees

The Presentation Visual Editor user interface will be an application for designing psychological experiments. It incorporates time-based decision trees in order to explicitly model time constraints and experiment structures visually. Therefore, this section will briefly introduce decision trees and time-based decision trees.

Decision trees are graph representations of decision rule systems

In general, decision trees are graph representations of hierarchical systems of consecutive decision rules. They are very common in domains such as stochastic, decision analysis, and artificial intelligence. Decision trees are used to facilitate decision making and to make less errors in the process. Russell and Norvig [2003] emphasize that decision tree representations seem to be very natural for humans and refer to manuals that are written as decision trees. They say that decision trees are used in a multitude of commercial applications, for example for financial decision making, and are the first classification method tried, when extracting decisions from data sets.

Decision trees are natural for humans

Decisions are made by performing a sequence of tests

Russell and Norvig [2003] say in the context of decision tree learning that "a decision tree takes as input an object or situation described by a set of attributes and returns a 'decision'—the predicted output value of the input." Decisions are made by performing a sequence of tests. Decision trees start with a branching root. Each internal node is a test of the value of one of the attributes. The outgoing branches are labeled with the possible values of the test that determine a decision, for example probabilities. Finally, the leaves specify the decision value that will be returned. These leaf nodes are reached through unique paths. In this way, the sequence of decisions yields a rule that determines a final decision. The rule can be easily read when traversing the tree from the root to the particular leaf. Hence, decision trees classify data into different groups that are each determined by a certain rule. An example for a decision tree can be seen in figure 2.1.

Decision trees classify data

Decision trees can be induced from example sets in machine learning

In machine learning, decision trees are usually generated top down from a set of examples. In each step the attribute is searched that discriminates the set best. The goal is to

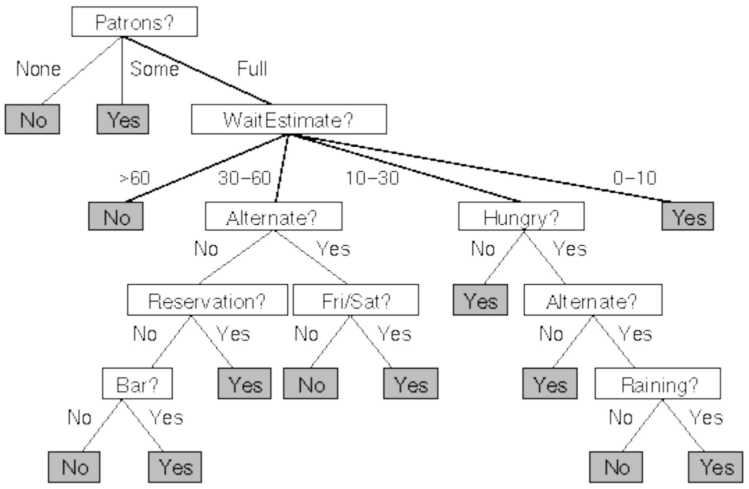


Figure 2.1: A decision tree for deciding whether to wait for a table in a restaurant (from [Russell and Norvig, 2003]).

classify the example data in such a way that the resulting decision tree is the smallest one consistent with the examples. This is done by testing the attributes and choosing the one that splits the set best. A split is good, if the resulting classification is as pure as possible, i. e. it distinguishes the examples best with regard to their decision goal. As a means information theory is applied. It allows computing the attribute with the highest information gain. This attribute then is chosen in the tree. The remaining attributes are examined in the next step of the algorithm to further classify the data. This is repeated until no further split is possible and the leaves represent classifications and the branches represent conjunctions of features that lead to those classifications.

Best attributes for building the tree are computed

In conclusion, decision trees have the important property, that it is possible for humans to understand the output [Russell and Norvig, 2003]. This point of view is shared by Eisenstein and Puerta [2000]. They state that decision trees are extremely readable and that their structure makes it easy to predict their effects. This is a clear advantage over other decision systems, such as knowledge-bases of rules and neural networks, which do not share these properties.

Humans understand decision trees: Great readability

2.2.1 Time-Based Decision Trees

Extension of decision trees: Regards time flow and time constraints

Traditional decision trees do not incorporate time for making decisions. Time-based decision trees, however, extend that notion of decision trees by including time. Decisions made by the tree now depend, in addition to the traditional tests, on time constraints and time flow. This increases the discriminatory power in systems that depend on time. Now the nodes can have an extra temporal label which specifies when a condition should be checked in order to select one of the branches or to make a decision. Additionally it is possible that a time condition is the only attribute that is tested in a node. Taking time constraints into account leads to additional properties when applying decision trees in software systems. For instance, the time needed by users for making a decision could affect the resulting decision of the tree. Furthermore, the tree must obey to the time flow. This has the effect that certain decisions can only be made at certain points in time as the system reaches different states in the course of time.

2.3 Timed Automata

Timed automata extend finite automata with including time constraints

An alternative model to time-based decision trees for including time constraints is the theoretical concept of timed automata. A timed automaton is an extension of a finite automaton. The state-transition graphs are annotated with timing constraints using finitely many real-valued clocks to model and analyze the behavior of real-time systems over time. A timed automaton accepts timed words, infinite sequences in which a real-valued time is associated with each symbol [Alur and Dill, 1994].

Synchronous clocks are assigned to automata

Timed automata consist of states and discrete transitions between them. Furthermore, clocks are assigned to the automata. The transitions correspond to events and take no time. The clocks are initialized with zero when the system starts and then increase synchronously at the same rate.

Clock constraints restrict behavior

Clock constraints on state transitions are used to restrict the behavior of the automaton. A transition, represented by an edge in the graph, can be taken when the clock values satisfy the constraint labeled on the edge. Clocks may be reset to zero when a transition is taken. A configuration of an automaton consists of the current state and the current value assignments of the clocks. There exist two kinds of state transitions: Timed steps after a time unit has passed and discrete transitions after events. In the case of a timed step, the state stays the same, but the clock values are uniformly increased. In the case of a discrete transition, the state changes accordingly to the edges in the graph. An example for a timed automaton is shown in figure 2.2.

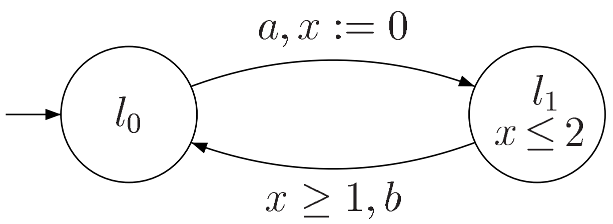


Figure 2.2: Timed automaton with single clock x: The automaton starts in state l_0 and moves to l_1 when reading a. The annotation $x := 0$ resets the clock when the edge is traversed. An annotation of form $x \geq 1$ on an edge gives the clock constraint associated with the edge. The transition from l_1 to l_0 is only taken, if the value of x is between 1 and 2. The time interval between the events a and b is always between one and two time units (from uni-ulm.de[1]).

Automata are related to formal language theory

However, state automata are more formal than decision trees as they are closely related to formal language theory and are based on a mathematical model. So, timed automata are studied most often from the perspective of formal language theory to consider properties such as closure. Thus their focus is different from that of decision trees. They concentrate on the classification of abstract problems according to their theoretical solvability. Additionally, timed automata are more complex to read and understand for humans than decision trees. These facts make timed automata less suitable for the application in this work and hence, time-based decision trees are preferred and used for the Presentation Visual Editor UI.

Different focus than decision trees

[1]http://www.informatik.uni-ulm.de/ki/Edu/Vorlesungen/Modellierung.und.Verifikation/SS07/folien06a.pdf

2.4 Time Constraints

Time is important in the design of interactive systems

Time aspects are an important element in the design of interactive systems. However, the explicit consideration of time issues and their effect on the interaction flow is often neglected by software developers during the design process due to the absence of appropriate tools. But a bad designed time flow in the user interaction forms an obstacle for the use of an application. The whole user experience can suffer from that.

Consider UI time

Here, time is not considered under performance issues, i. e. for this work it is irrelevant how fast systems are or how high their computing power is. Instead it is important to consider the UI time: How does the system react to the user, how far are time factors considered, and what is the temporal design of the interaction?

2.4.1 Time Constraints in Human Performance

Time is important factor in human behavior

Time is an important factor in human behavior. Card et al. [1983] introduce the basic concepts and times of human cognition and information processing to the field of human-computer interaction (HCI) with their "Model Human Processor". It should be mentioned that this is only a simplified psychological model that does not give a whole theory of human cognition. This model, also called the CMN model, is depicted in figure 2.3. It presents a basic model for perception, cognition, and motor system using three human processors and their associated memory. Its goal is to estimate execution time, error rates, and training effects for simple input/output events. The Model Human Processor illustrates, that reaction times and time constraints in human memory performance are of great importance when designing interactive systems that should adhere to human capabilities. Apart from pure performance issues, these concepts also give valuable insights for general UI time constraints that should be respected throughout the design.

CMN model: Human cognition and information processing

Three human deadlines

The CMN model is, for example, the basis for the three hu-

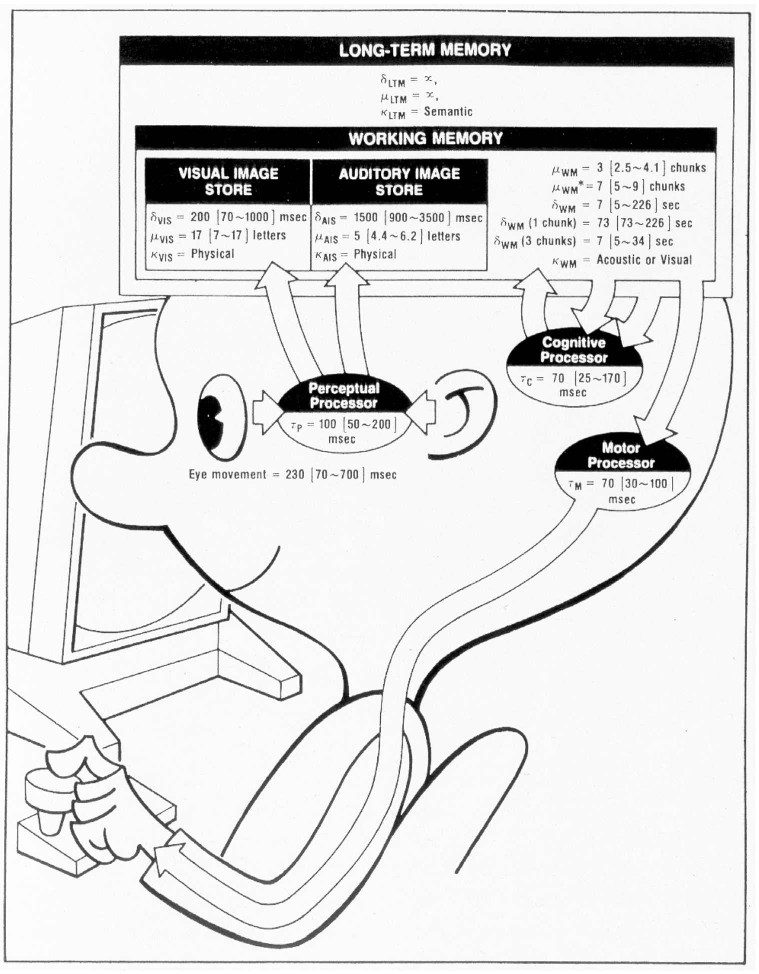


Figure 2.3: The Model Human Processor (CMN model) illustrating three human processors of information processing and their associated memory (from [Card et al., 1983]).

man deadlines described by Johnson [2007]. The deadlines identify human limits for attention spans (10 seconds), recommended times for giving feedback (1 second), and, directly derived from the CMN model, the time for causality breakdowns in the perception of cause and effect (0.1 seconds). These are some basic human time constraints that must be regarded in the design of interactions. Thus, the deadlines indicate how long one task step should last at maximum or when progress indicators should be displayed.

Time constraints in interaction design

Golden rule: Minimize memory load

But the CMN model is also the foundation for many further time constraints in human performance. For instance, one of the golden rules of interface design, defined by Nielsen [1994], is to minimize memory load. Here, the fact should be exploited that a user can remember approximately seven chunks (information items) in short-term memory. However, the storage time in working memory for these chunks is short. Thus, there is a possible need for reminders when a user has to remember several items over time because of the short storage time. An even better solution could be to design the interaction in such a way that there is no need for the user to remember things over time.

Short storage time in working memory

2.4.2 Time Constraints in Interactive Systems

Time issues on web pages

In order to show the importance of time, in this section, I will give some examples for interactive systems in which time issues arise. First, traditional web pages often have interaction problems. These pages are usually static and hence cannot provide appropriate feedback to the user. A site can only react to user input by loading a whole new page. The resulting reaction times can disturb the interaction. In addition, a site cannot initiate actions, that would be necessary for a good interaction, by itself. These are clear disadvantages to the possibilities desktop applications can offer. Modern web technologies, such as AJAX, can overcome these limitations since they can provide functionality comparable to common desktop applications. So, nowadays it is possible to design web pages with improved time lapse.

Initial state problem for ubiquitous computing devices

In the emerging field of ubiquitous computing problems concerning time lapse can occur. A fundamental problem is, for example, the initial state that a user discovers when using a device. Since ubiquitous devices should not be personal devices anymore but publicly available, the question arises what will happen to the system state when a user stops interacting with device. It is important to consider whether the system can take actions over time to facilitate access for the next user.

State problem in interactive exhibits pattern language

The state problem can be further explored when considering the interactive exhibits pattern language described by Borchers [2001]. The pattern "H4 Easy Handover" defines this problem as follows: "Most interactive systems implicitly assume that each user begins using their system from a start page or initial state. At interactive exhibits, however, one user often takes over from the previous one, possibly in the middle of the interaction, and without knowing the interaction history of the previous user." In the case that the system was not returned to the initial state of interaction, the suggested solution is: "Minimize the dialog history that a new user needs to know to begin using an interactive exhibit. Offer a simple means to return the system to its initial state."

User can leave exhibit in the middle of interaction

Time constraints for interactive exhibits

In the aforementioned situation also issues concerning time constraints need to be solved. If a user leaves the exhibit in the middle of the interaction and no successor is already waiting there, what will happen? Will the system stay in the current state until the next user takes over? Will it return after some time to the initial state? The problem is that it cannot generally be assumed that the user has already knowledge of how to use the system and therefore he might not be able to get back to the start page explaining the system. This is especially true for exhibits with non-standard input devices. In "Easy Handover" this is not regarded, as it is stated: "The new user cannot be expected to have seen the introductory screen explaining how the batons are used. This knowledge, however, is something that the new user has already gathered from watching his predecessor using the exhibit before taking over." If he has not obtained this knowledge, good temporal design of the system can facilitate smooth interaction. These problems can arise with many public systems.

How to attract users

Another issue in the context of public spaces are interactive systems that should attract users. While trying to allure potential users, the system must not annoy passers-by. Here again, time constraints can be used as a valuable tool, for example for sending time controlled stimuli.

Time in communication systems

In private systems also many interesting temporal interaction questions can arise. For example, when considering communication with instant messengers: If a user does not respond to his conversational partner for some time, should the system inform the other person that the user is busy? Maybe the system has monitored no activities at all from the user for some time and could even decide to inform the other person that the user is away.

The last example is about general interaction. If a user tries to do so some settings in a configuration system, but goes back and forth without making progress, maybe it is too confusing and he should be offered help by the system?

All examples in this section had the purpose to make clear how important time considerations in interactive systems are and that software development tools should support the modeling of interaction time.

2.4.3 Time Constraints in Software Development

In order to be able to differentiate a design including time-based decision trees from common techniques used today, this section will look at current design and software development processes, in particular at the visualization method Unified Modeling Language (UML) [OMG, 2007].

Focus is on spatial layout; not on temporal layout

Design interactions, not just interfaces

Time only used for visualization of internal processes and architecture

The general focus of software development tools is mainly on the spatial layout of UIs. Many integrated development environments (IDEs) provide excellent support for visually defining the spatial layout. Comparable means for specifying the temporal layout do not exist. But when designing interactive systems, not only interfaces have to be designed, but also *interactions*. At most, prototyping tools like Adobe Flash[2] offer functionality for temporal design as Flash implements a time line metaphor. This application is, however, intended for creating animations and web content and is in general not suited for building whole applications. Current software modeling techniques, like UML diagrams, use time only for the visualization of internal processes and system architecture. Thus, the focus is on system time and those techniques all disregard UI time constraints. Since there is no tool support for time constraints, they are hardly considered by developers during the design process.

[2]http://www.adobe.com/products/flash/

In the following, UML will be covered as an example for a software technique. UML contains 13 types of diagrams, which can be divided into structure diagrams, e. g. class diagrams, and behavior diagrams, e. g. activity and use case diagrams. Behavior diagrams illustrate the dynamic behavior of a system by showing collaborations among objects and changes to the internal states of objects. Interaction diagrams are a subtype of behavior diagrams and illustrate the flow of control and data among objects in the system.

UML contains 13 diagram types

In the context of time, sequence and timing diagrams are of most importance. These are specific types of interaction diagrams with the focus on timing constraints. They specify temporal behavior more precisely than other UML diagrams and are therefore better suited for the design of real time systems. Both diagrams show the dynamic behavior of objects throughout time. Sequence diagrams show, as parallel vertical lines, different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. Time increases from top to bottom. An example can be seen in figure 2.4. Timing diagrams are a special form of sequence diagrams. They are two-dimensional, too, but with time increasing from left to right and the ordinate showing the object state.

Sequence and timing diagrams have focus on timing constraints

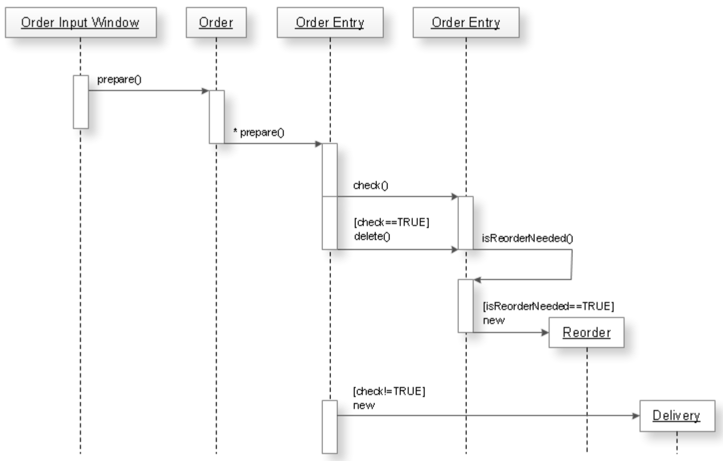


Figure 2.4: Example for a UML sequence diagram.

Illustrate system time, not UI time

As pointed out in this section, these diagrams show internal processes and system architecture rather than interaction objects. They illustrate the system time, i. e. how the processes behave over time, and do not show UI time constraints, i. e. they are not used for visualizing the interaction flow.

2.5 Iterative User-Centered Design

Systems become more usable when users are involved in the design process

User-centered design is a design process that aims at identifying and meeting the requirements, wants, and limitations of the end-users of a computer product or computer interface at each stage of the development process. Therefore, users have to be involved in the entire design process. The goal is to create more successful systems with an enhanced usability.

UIs should be designed iteratively

Impossible to create perfect design in single attempt

Nielsen [1993] extends this process with introducing iterative user-centered design. He says that "user interfaces should be designed iteratively in almost all cases because it is virtually impossible to design a user interface that has no usability problems from the start. Even the best usability experts cannot design perfect user interfaces in a single attempt, so a usability engineering lifecycle should be built around the concept of iteration."

DIA cycle characterizes iterative design process

This iterative design process is characterized by the design-implement-analyze (DIA) cycle (see figure 2.5). In the design phase, a design is created from concepts and ideas and possible prior analyses. If a prior iteration has preceded this design phase, its findings from the analysis phase are used for refining the existing design. In the implementation phase, this design then is implemented. This can result in early low-fidelity prototypes, that aim at fundamental high-level feedback, in subsequent high-fidelity prototypes, that aim at detailed feedback, or in the final system. Afterwards, in the analysis phase, the system is evaluated in usability tests. Here feedback of potential users about the system and the design is collected in different forms of user tests and surveys. With each iteration of the DIA cycle, the design becomes more concrete and precise and the implementation becomes more usable and technically complex. Thus, the user feedback in the analysis phase focuses on smaller and smaller problems in the system.

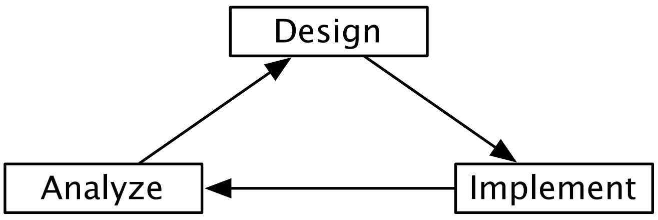


Figure 2.5: The DIA cycle.

This work adheres to the iterative user-centered design process

Throughout this thesis, my work has adhered to the iterative user-centered design process. Before I ran through the first iteration, I began with an extra analysis phase where I have conducted a Contextual Inquiry to identify current work practices and problems and to get a deep understanding of the psychological domain. Based on the obtained results I have designed, implemented, and evaluated three prototypes, two paper prototypes and one interactive software prototype. In the design phases I have applied the analysis results of the previous cycle for further refining my design. Some results, however, could not have been incorporated in the designs, yet, and are treated in the future work section. Finally, I have performed an additional evaluation to test my system in more detail and to compare it with the original Presentation interface. In this way, I was able to obtain not only qualitative results, as in the DIA evaluations, but also quantitative measures for the quality of my interface.

2.6 Designing for Domain-Expert Users

Domain-expert users have special demands

When designing software for domain-expert users, such as psychologists, special demands should be taken into consideration. Costabile et al. [2003] define domain-expert users as a specific category of computer end-users who are "experts in a specific domain, different from computer science, who need to use computers to perform their

Domain-experts are no computer scientists

daily work tasks". The authors emphasize that these users are not and do not want to become computer scientists. The challenge for designers of computer systems more accessible to their end-users is to develop programming paradigms and software environments that are adequate to the needs of end-users. Otherwise, Raskin's laws of interface design which say that a computer should neither harm your work nor waste your time [Raskin, 2000] cannot be observed as various problems will occur throughout the domain-expert users' daily work.

Co-evolution of users and systems

Nielsen [1994] said that "using the system changes the users, and as they change they will use the system in new ways". Costabile et al. [2003] concluded that the designer must in turn evolve the system to adapt it to its new usages, which they called co-evolution of users and systems. They identify user creativity, i. e. devising new ways to exploit the system to satisfy their needs, and user acquired habits, i. e. accustomed interaction strategies that should be facilitated, as the two main reasons for co-evolution.

Domain-experts perform end-user development

Domain-experts often feel the need to perform various programming activities—despite having little or no programming knowledge. That may even lead to the creation or modification of software artifacts, in order to get a better support to their specific tasks, thus being considered activities of end-user development. The required kind of programming is rather simple most of the time, but as existing software does not usually provide any programming capability, users are confronted with difficulties.

Programming to save time and to add features

The need for programming activities in end-user software was also shown by Dorn and Guzdial [2006]. They showed domain-expert users' needs by the example of graphic designers. The authors observed that the characteristics and behaviors in this domain are similar to those in other end-user domains. They found out that end-users, despite having no or only little formal computer science education, are taking part in significant programming activities. As end-user programmers they make use of features like textual scripting or automation. Programming serves as a means to save time and add features to their applications. In order to accomplish their tasks, the graphic designers borrow code from examples and rely on documentation. They even appear to be discovering knowledge that is common in formal computer science learning environments.

Standard tools become insufficient for non-standard goals

Dorn and Guzdial [2006] discovered in their study that learning to program is a natural progression for end-users since the standard affordances of their tools become insufficient as users have more and more sophisticated needs. Programming enables them to build enhanced graphic effects and to save a lot of manual work. Thus, there is a natural demand for programming capabilities in software applications when users want to achieve non-standard goals that should be provided by the software. Otherwise, users may become frustrated because they need much time or are not able to fulfill their tasks.

Chapter 3

Related Work

"Research is to see what everybody else has seen,
and to think what nobody else has thought."

—Albert Szent-Gyorgyi

Time-based decision trees are not used in interaction design

Time-based decision trees have not been applied in the context of interaction design yet. However, there are several software systems that take advantage of decision trees. In this chapter, I will therefore present a small selection of two examples that show how decision trees—with and without time conditions—are used in software. Furthermore, I will introduce alternative approaches to Presentation to present stimuli and to design psychological experiments. I will argue their characteristics, their similarities and differences to Presentation, and why the Presentation Visual Editor provides superior functionality.

3.1 Decision Trees in Automated User-Interface Design

Decision trees used to support automated UI design

Eisenstein and Puerta [2000] use decision trees to support automated UI design by providing an adaptive algorithm that recommends an ordered list of interactors to the designer. They do not consider time conditions in their work.

Design is hard to automate since success is based on flexible standards

The application takes advantage of the flexibility and the comprehensibility of decision trees. Since design problems demand creative solution processes, human designers often proceed by intuition and are unaware of following any strict rule-based procedures when they make design decisions. Furthermore, design success is often very subjective and depends on stylistic preference and flexible standards. There is no objective scale of utility of a design, so that two designers can make very different decisions without either being wrong. While there is often not the one *correct* design, some decisions will lead to bad designs. These properties make design tasks especially difficult to automate. Eisenstein and Puerta [2000] propose adaptation as a means to overcome these challenges.

Adaptive automation of UI design

They aim at automating user interface design adaptively, so that the software takes the designer's personal preferences into account and is able to learn fast how to handle new UI widgets and technical innovations. They point out that interface researchers will also benefit. The results of using the adaptive algorithm allow discovering information about the way designers make decisions.

Automation must be comprehensible

It is of great importance to the authors that the automation algorithm is comprehensible to the interface designer, i. e. the user understands how and why automatic decisions are made, and that the designer is able to influence decisions explicitly and directly.

Selection of interactors is automated

Eisenstein and Puerta [2000] do not aim at creating the entire UI automatically, but focus on automating those features that reasonable benefit from automation. In their work, they handle the selection of interactors, visual elements, such as buttons or sliders. Their automation algorithm returns an ordered list of interactors from which the interface designer then chooses while performing layout. They hope to further increase the number of automatic design decisions with further research.

Decision trees are extremely readable

In order to perform those automatic mappings of interactors and interface elements, a decision tree is used. The authors chose decision trees, because they are extremely readable and their structure makes it easy to predict their effects.

In this system, the decision tree classifies objects that the user of the interface will modify or view based on discriminants and assigns a set of possible interactors based on this classification. Discriminants are features of the cases that may be relevant to how they are sorted. The decision tree specifies which discriminants to consider and in what order. In essence, mappings are created between domain objects and interactors.

Designer gets ordered list of interactor recommendations

The application uses five discriminants. For example, one discriminant is the type of a domain element. It would not make sense to assign an edit field to a Boolean type. A better choice would be a check box or a radio button. Another discriminant uses the number of allowed values to decide whether a set of radio buttons, a list box, or a drop-list is the recommended interactor. An example decision tree for these discriminants is depicted in figure3.1. Using these principles, an ordered list of recommendations for the interface designer is produced.

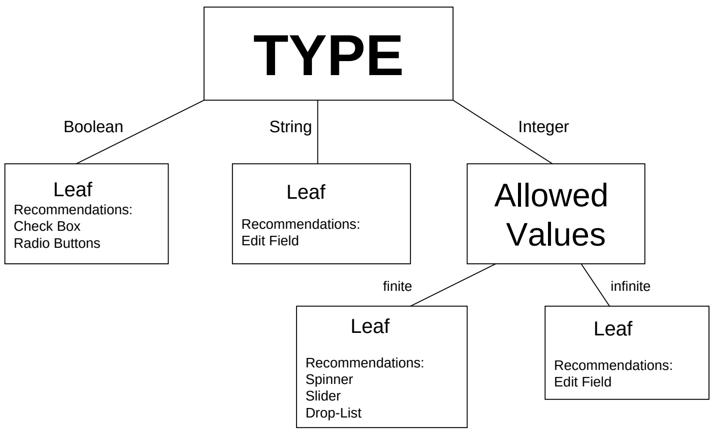


Figure 3.1: Example for a simple decision tree for interactor selection.

Adapt decision tree to designer's preferences

The designer can always correct the automatic interactor recommendations. Whenever the designer does so, the interaction is recorded as an error, otherwise it is considered successful. After each session an adaptive algorithm is applied to correct the decision tree, regarding the entire history of cases. Preliminary experiments showed that their algorithm worked, i. e. the decision trees produced less errors after adaptation, and that the performance of the designers was improved significantly.

This system is an example for how the characteristics of decision trees, such as understandability, can be used to improve a creative design task. Although the approach of this work is completely different from the work in this thesis, the advantages and the power of decision trees as a means for design tasks in software become clear.

3.2 Time-based Decision Trees for the Diagnosis of Embedded Systems

Time-based decision trees for on-board diagnosis in dynamic embedded systems

The next work, I will present, considers time conditions in the context of decision trees. Console et al. [2003] introduce the notion of time-based decision trees as a model for on-board diagnosis in dynamic embedded systems. They show that diagnostic decision trees are an efficient model for reasoning about appropriate recovery actions in response to system faults. Then the trees are extended to time-based decision trees to exploit temporal information about observations and to preserve time constraints.

Embedded systems must be monitored to be reliable

The basis for this work is that the embedding of software components inside physical systems is widespread today. These systems must behave properly and fault-safe, and they must guarantee a high availability. Furthermore, systems are often safety critical, e. g. a car braking system. Thus, the monitoring of system behavior, the detection and isolation of failures, and the performance of appropriate recovery actions is a critical task of control software.

Designing on-board diagnostic software is hard

The design of diagnostic software, however, is complex, expensive, and time consuming. Problems get especially hard, when designing on-board diagnostic software, since on-board resources, such as memory and computing power, must be very limited to keep costs low. Nevertheless, near real time performance is needed. At the same time, the costs for diagnosis must be kept acceptable, e. g. in terms of the number of sensors to monitor the system. An aggravating factor is that the devices often have dynamic and time-varying behavior and the whole system is complex and interdependent.

Decision trees as efficient model for reasoning

Console et al. [2003] improve the efficiency through automation with model-based reasoning. They choose decision trees as a model, because they are efficient in time and space and they can be used as a comprehensive and compact representation of the system behavior. Further, there exist well established machine learning algorithms like ID3 [Quinlan, 1986] for generating decision trees automatically from a set of examples. In this case, the examples to be learned are pairs of diagnoses and recovery actions. The decision trees can be used to implement classification problem solving and thus some form of diagnostic procedure.

Time-based decision trees for coping with temporal system behavior

Basic decision trees have the drawback that they cannot properly handle the temporal behavior of the systems to be diagnosed. They are not capable of making decisions on actions based on observations that were acquired across time. Therefore, traditional decision trees are extended to time-based decision trees. Now the nodes have a temporal label which specifies when a condition should be checked in order to select one of the branches or to make a decision. This leads to an improved discriminatory power in the model of a dynamic system. It can be taken into account that data may be observable at different times and that temporal patterns of data may be the only way of distinguishing different faults. An example time-based decision tree for diagnosis can be seen in figure 3.2.

Temporal information has to be exploited

Thus, neglecting the notion of time may limit the decision process. In order to optimally discriminate errors temporal information has to be exploited. The strategies can be versatile. “In some cases there is nothing better than waiting” to get data for a good and distinct discrimination, but this is not always possible as the safety and integrity of the physical system must be maintained. It is important to have deadlines for performing recovery actions that always have to be met. This can be achieved by required time constraints in the tree.

Nodes contain time labels, leaves are labeled with recovery actions

The resulting formal approach generates monotone trees in which time labels do not decrease when moving from the root to the leaves of the tree. The nodes consist of a

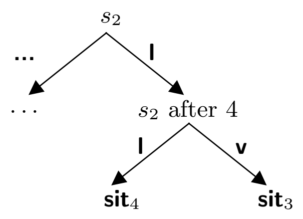


Figure 3.2: Example for a diagnostic time-based decision tree: sit_3 and sit_4 are fault situations that need to be distinguished by sensor s_2. Both can be detected by s_2 showing a **l**ow value that later turns into **v**ery low. The faults can only be discriminated by their temporal pattern: sit_3 reaches value **v** after 4 time units, sit_4 after 6 units. Thus, a decision tree incorporating time can make a decision 4 time units after value **l** was detected.

test and a time label for choosing the next child to traverse. The leaves are labeled with decisions, i. e. recovery actions. Then these diagnostic trees have to be used when some abnormal value is detected for some sensor. With a variation of the ID3-algorithm that, includes example sets with temporal information, the temporal diagnostic decision trees can be generated automatically.

System depends on time constraints

The work of Console et al. [2003] gives an example for systems that really depend on time constraints and shows that time-based decision trees have the capabilities to solve such problems efficiently. The differences to the Presentation Visual Editor in the way of applying time-based decision trees are huge, though. The trees are neither visible nor are they an element of interaction as the whole system is not interactive at all. Further, machine learning algorithms generate the trees automatically from example sets whereas in the Presentation Visual Editor the trees are generated by users.

Huge differences to the Presentation Visual Editor

Both system benefit from time-based decision trees

Even though there are fundamental differences between diagnostic procedures and creative tasks like experiment design, both systems benefit from the feature of representing timed decision structures comprehensible and compact.

3.3 SMIL

In Presentation, the Scenario Description Language (SDL) is a descriptive language used to specify stimuli and sequences of stimuli and their associated timing and layout properties. For more details, see chapter 4.1. A language that is related in some way to SDL, is the Synchronized Multimedia Integration Language (SMIL) [W3C, 2005], which was first released in June 1998. SMIL is a standard of the World Wide Web Consortium (W3C) for a markup language for integrating and controlling multimedia contents in web pages that specifies timing, layout and synchronization. It is based on the Extensible Markup Language (XML).

SMIL is an XML-based markup language

SMIL is used to describe multimedia presentations in web pages and provides more functionality than the Hypertext Markup Language (HTML). HTML provides support for exactly defining the layout of web pages and is able to integrate objects of different formats and to create both static and dynamic presentations. But HTML cannot control the time flow of the transmission or of the presentation of the individual objects. Multimedia presentations, however, need exact time control. For that purpose, SMIL was developed. It can control the time flow of presentations as well as their layout and serves to position, synchronize, and present multiple multimedia objects, such as audio, video, texts, and graphics.

SMIL describes multimedia presentations on web pages

Controls timing and layout

SMIL can be linked with Java-applets. The presentations can be adjusted to bandwidth, color depth, and screen resolution. SMIL is even implemented on mobile devices and can be used for Multimedia Messaging Service (MMS).

Hence, SMIL is a powerful language for audiovisual presentations and has been described as the Internet answer to Microsoft PowerPoint. It is rather easy to learn reading and writing SMIL documents as they are based on XML and use its syntax. An example for SMIL can be seen in listing 3.1. The structure of SMIL and SDL documents is similar in some points as "stimuli" are defined together with their position and timing parameters. However, their intended use is very different what makes a direct comparison difficult.

Internet answer to PowerPoint

```
<smil>
 <head>
  <layout>
   <root-layout width="300" height="200"
        background-color="white" />
   <region id="pic" left="75" top="50"
        width="116" height="81" fit="fill"/>
  </layout>
 </head>
 <body>
  <img src="pic.jpg" alt="Pic" region="pic"
       dur="3s" begin="2s" />
 </body>
</smil>
```

Listing 3.1: SMIL example: Two seconds after starting the presentation a graphic file is displayed for three seconds (from hdm-stuttgart.de[1]).

SMIL not suitable for experiments

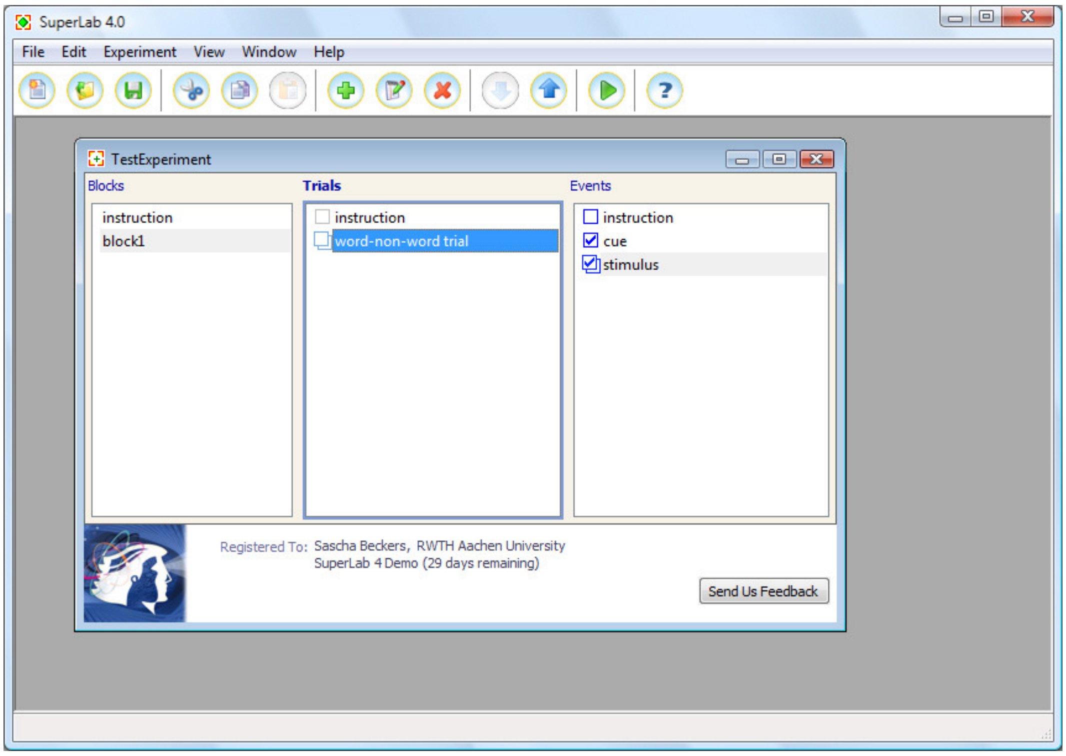
SMIL is not as powerful as SDL, as it is intended for designing slide shows and Internet presentations that are merely linear. It would not be possible to implement simple psychological experiments with SMIL because experiments require much stricter and more precise timing conditions. Additionally, SMIL provides no means for data recording and analysis.

3.4 SuperLab

SuperLab is GUI-based stimulus presentation software

The company Cedrus [2008] developed SuperLab 4.0, a stimulus presentation software, that provides support for playing movies as well as for various graphic file types. SuperLab is a competing software to Presentation for realizing psychological experiments. The application has a GUI and the developers stress that no programming is needed at any point when designing experiments. This is a strong contrast to Presentation, where everything is based on scripting languages (see chapter 4.1).

[1]http://www.hdm-stuttgart.de/streamingmedia/SMILTextbuch/Zeitvier.htm



Blocks	Trials	Events
instruction	☐ instruction	☐ instruction
block1	☐ word-non-word trial	☑ cue
		☑ stimulus

Registered To: Sascha Beckers, RWTH Aachen University
SuperLab 4 Demo (29 days remaining)

Figure 3.3: Screenshot of SuperLab showing the main screen with an experiment

Figure 3.3 displays the main screen of SuperLab containing a small experiment. Experiments are structured into blocks, trials, and events. In each section the user can define elements, e. g. an event, that will display a picture. Then the user can link blocks, trials, and events with check boxes in order to specify which events are contained in which trial and which trials are contained in which block. This has the advantage that events and trials can be reused very easily in other parts of the experiments. But at any time it is only possible to overview one trial or one block. There is no possibility to overview the entire experiment structure at once.

Nested dialog windows

There are some drawbacks in the operation of SuperLab. For example, there is a high number of nested editor windows: an event editor, a trial editor, and a block editor. Each dialog window has a couple of tabs which might invoke additional windows. In this way, there is no at a glance overview of the settings for experiment elements and events. At any time only parts of the structure are visi-

Unnecessary complicated functions

ble what makes it difficult to get an idea of the structure. In addition, some standard functions seem to be unnecessarily complicated. For instance, simply displaying four pictures simultaneously requires creating four picture events. The first three pictures must be kept invisible, the last three must not erase the screen and then making the last one visible causes all four to appear at the same time—after they have all been linked to one trial.

Abstract time management

The principles of defining time structures are similar to Presentation. Thus, time management is rather abstract. As displayed in figure 3.4, the settings are quite hidden although timing is of great importance in experiments.

Conditional experiment branching possible

SuperLab has capabilities for designing experiments with conditional branching. Since this can be achieved without programming, branched experiments can be implemented easier than with Presentation—with the restriction that branched structures are not directly visible but hidden in the editor dialogs, too. Hence, some problems in experiment design and management persist although having a graphical interface that would allow for more clarity and visibility.

No visual representation of structures

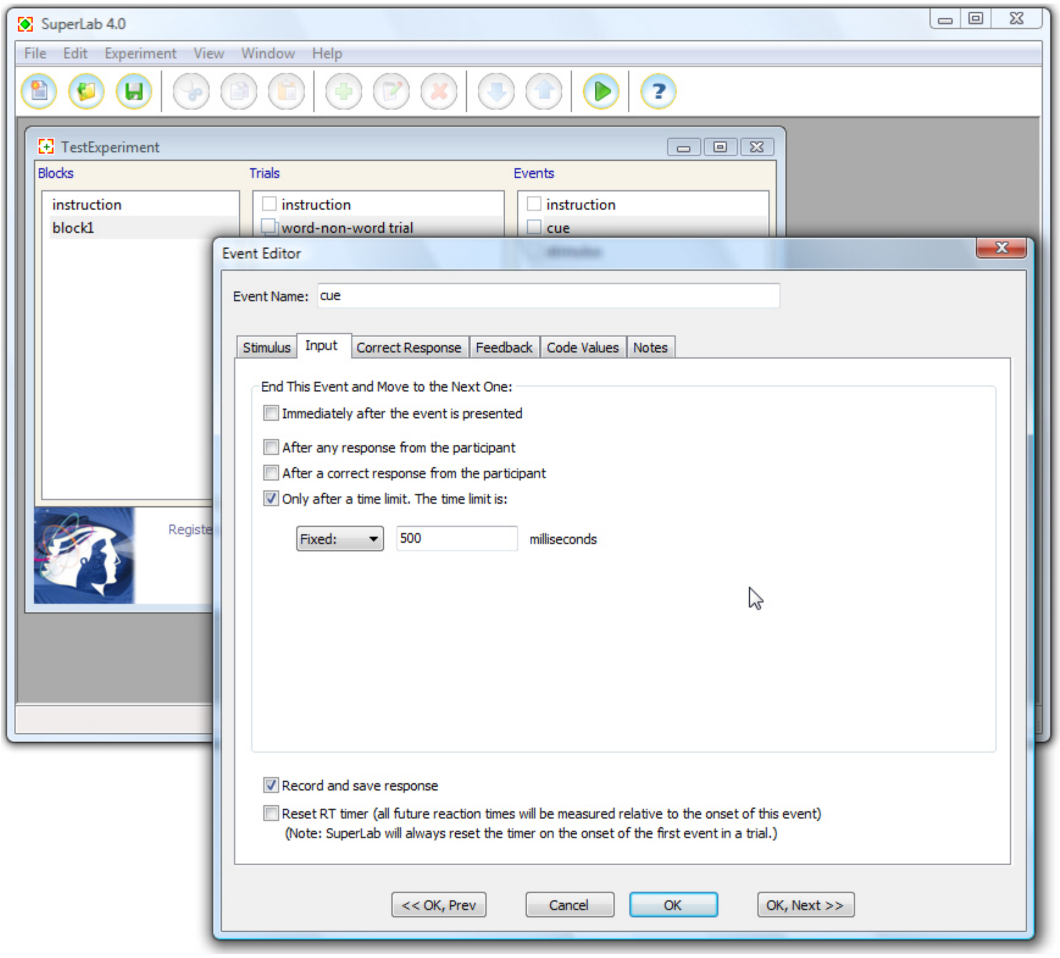
As in the Presentation Visual Editor the whole interaction is visual. But SuperLab neither supports direct manipulation of stimuli and layouts nor is the experiment structure displayed visually. There are no functions for representing time visually. Instead there are only abstract settings for stimulus event durations and response dependent durations; the course of events cannot be seen directly.

Less powerful than Presentation

All in all, SuperLab is less powerful than Presentation. There are less features, less supported file types, less stimulus types, and less supported devices. This application is not suitable for the kind of research the Department of Psychiatry and Psychotherapy conducts as there is only constrained fMRI support and because timing is not as precise as needed.

Lack of programming capabilities limits users

SuperLab has the great advantage of visual interaction without programming, but users are constrained in comparison to Presentation. Presentation allows programming complex experiment conditions. This offers more possibili-



Event Name: cue

End This Event and Move to the Next One:
- [ ] Immediately after the event is presented
- [ ] After any response from the participant
- [ ] After a correct response from the participant
- [x] Only after a time limit. The time limit is:

Fixed: 500 milliseconds

- [x] Record and save response
- [ ] Reset RT timer (all future reaction times will be measured relative to the onset of this event)
(Note: SuperLab will always reset the timer on the onset of the first event in a trial.)

Figure 3.4: Screenshot of SuperLab showing the event editor tab for setting time parameters.

ties and flexibility. Throughout the evaluations of my prototypes, I asked the participants whether they had gained experience with SuperLab. One researcher stated, that he was not able to realize a complex experiment with SuperLab because of the lack of programming capabilities. One solution to this problem would be to introduce—in addition to the graphical interface—end-user programming for adjusting the software to advanced needs.

Presentation Visual Editor offers programming capabilities

Chapter 2.6 already described that such constraints are likely to occur when there is no possibility for domain-experts to do end-user programming. Consequently, it seems important that the Presentation Visual Editor is not merely a graphical experiment interface but is built on top of Presentation's scripting languages and is designed to output Presentation scripts. Thus, it is ensured that users are not constrained as they can use Presentation's languages to add enhanced functionality for specific non-standard tasks. These optional programming capabilities will prevent that the Presentation Visual Editor will experience the same problems as SuperLab in the design of complex experiments.

Finally, in another interview about SuperLab, one researcher said that he misses clear visual overview of experiments and time management after having seen the Presentation Visual Editor in comparison.

Chapter 4

Current Work Methods

"Pleasure in the job puts perfection in the work."

—Aristotle

In order to test the idea of applying time-based decision trees in interaction design, we chose the highly specialized domain of psychological research. The reason for this choice was that time aspects are of great importance in psychological experiments. Thus, this domain could be an excellent application area for taking advantage of time-based decision trees and demonstrating their benefits in interaction design. Before being able to start with designing a new experiment interface, first the current situation and the current used software have to be examined.

Test time-based decision trees in psychological domain

Time is important in psychological experiments

In this chapter, I will describe the current work conditions in the Department of Psychiatry and Psychotherapy when designing experiments. Therefore, I will describe the application Presentation which is used for implementing experiments and which will be the basis for my new user interface. Besides gathering theoretical knowledge about the functionality of Presentation, I conducted a Contextual Inquiry with users who were actively working with Presentation. The inquiry serves the purpose of learning what the users do with the software and how they do it. I wanted to find out about why something is done—or not done—and

Investigate current software and work conditions what problems occur in the process, i. e. why does it make sense to improve the program. In addition, the Contextual Inquiry serves as means to get an understanding of the domain.

4.1 The Software System "Presentation"

Stimulus delivery and experimental control software system

The program Presentation is developed by the company Neurobehavioral Systems [2008]. A screenshot of the program is shown in figure 4.1. Presentation is a stimulus delivery and experimental control software system for neuroscience used for designing, performing, and analyzing psychological and neurological experiments. This software is not only common in the University Hospital Aachen, but, as I found out in interviews with local researchers, it is also the standard experiment software in many psychological research facilities in Europe and America.

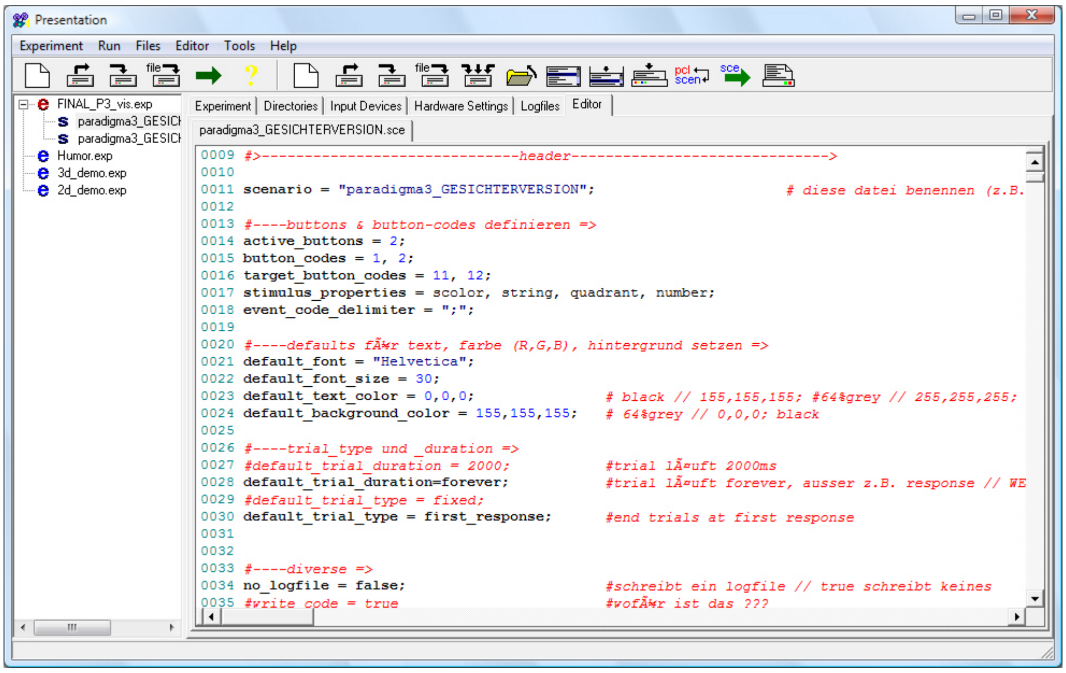
Great functionality

Presentation can deliver auditory, visual, and multi-modal stimuli with sub-millisecond temporal precision on standard hardware. Visual stimuli can be graphic files as well as animations, video files, and 2D/3D stimuli. It is possible to present multiple stimuli simultaneously. Presentation can monitor and record I/O ports and responses, such as button presses from mouses, keyboards, or joysticks, with high accuracy. Log files report the times of any event of interest. An interface allows controlling and triggering many external devices. For instance, it is possible to present stimuli on an olfactometer. Presentation was designed for behavioral and physiological experiments and provides support for fMRI scanners and other imaging techniques. It can receive fMRI pulses so that a scanner is able to trigger Presentation on a specific pulse. Its stimulus presentations can be controlled to be in time with the scanner pulses.

4.1.1 Scenario Specification

Textual specification of experiments

Presentation uses a text description to specify both the stimuli to present and how to present them. Experiment units


Presentation
Experiment Run Files Editor Tools Help
FINAL_P3_vis.exp
paradigma3_GESICH
paradigma3_GESICH
Humor.exp
3d_demo.exp
2d_demo.exp
Experiment | Directories | Input Devices | Hardware Settings | Logfiles | Editor
paradigma3_GESICHTERVERSION.sce

```
#>-----------------------------header------------------------------>

scenario = "paradigma3_GESICHTERVERSION";              # diese datei benennen (z.B.

#----buttons & button-codes definieren =>
active_buttons = 2;
button_codes = 1, 2;
target_button_codes = 11, 12;
stimulus_properties = scolor, string, quadrant, number;
event_code_delimiter = ";";

#----defaults fÃ¼r text, farbe (R,G,B), hintergrund setzen =>
default_font = "Helvetica";
default_font_size = 30;
default_text_color = 0,0,0;                  # black // 155,155,155; #64%grey // 255,255,255;
default_background_color = 155,155,155;      # 64%grey // 0,0,0; black

#----trial_type und _duration =>
#default_trial_duration = 2000;              #trial lÃ¤uft 2000ms
default_trial_duration=forever;              #trial lÃ¤uft forever, ausser z.B. response // WE
#default_trial_type = fixed;
default_trial_type = first_response;         #end trials at first response


#----diverse =>
no_logfile = false;                          #schreibt ein logfile // true schreibt keines
#write code = true                           #wofÃ¼r ist das ???
```

Figure 4.1: Screenshot of Presentation showing the built-in editor

are called scenarios. A scenario is a sequence of actions that Presentation performs. An experiment can consist of one or more scenarios whereas a scenario may correspond to one experimental condition or many. For specifying a scenario two languages are used. The Scenario Description Language (SDL) is a descriptive language used to specify stimuli and sequences of stimuli and their associated properties. The Presentation Control Language (PCL) is an interpreted programming language used to implement custom control of scenarios. SDL is required for writing a scenario, but it is not necessary to apply PCL.

Two languages: SDL and PCL

The Presentation developers compare scenario specifications with recipes that Presentation uses to cook an experiment. A recipe usually has two distinct sections: a list of ingredients, and a set of instructions. The former lists the things that will be used, but the latter actually tells what to do with those things. In this analogy, SDL is the list of ingredients, while PCL is the list of instructions.

Cooking recipe analogy

A scenario specification is stored in a scenario file that has three distinct parts. It starts with the scenario file header

Structure of scenario files

which contains definitions of various parameters that affect the scenario as a whole or that are used as defaults for parameters in the scenario. Next is the SDL part, which describes the stimuli and stimulus sequences that will be used in the scenario. The last (optional) part is the PCL program. It is also possible to separate one scenario description into multiple files. The PCL part can be in its own file and sections of SDL can be placed into separate template files that are referenced in the scenario file.

SDL

Scenario objects represent aspects of the stimulus delivery

The SDL part of a scenario consists of a series of statements that define the components of the stimuli to be used by the scenario. The whole section is constructed by using scenario objects. A scenario object represents some aspect of the stimulus delivery. There are, for example, picture, bitmap, sound, trial, and stimulus event scenario objects. They all have associated parameters. Scenario objects can be nested. For example, a picture object, which represents one full screen of graphics, can contain one or more picture part objects, such as bitmap or text objects.

Stimulus events describe particular presentations of stimuli

Trials represent stimulus sequences

Most scenario objects are related to stimuli. So, bitmap objects contain graphic data and wavefile objects contain sound data. The most important exceptions are trial objects and stimulus event objects. A stimulus event is one particular stimulus with associated parameters that describe one particular presentation of that stimulus. A trial object contains a list of stimulus events. Therefore, a trial represents a sequence of stimuli. An example of an SDL section is shown in listing 4.1.

Template files useful for repeated structures

The SDL code in a template file can be processed multiple times in a scenario. A template table in the scenario file specifies the number of iterations and the values that are passed to the template for each iteration. Since variables are often used in templates, the resulting code can vary each time it is processed. Thus, template files are useful for repeated structures with minor modifications. But the code could be placed as well one or more times in the scenario file. They are used to make writing a scenario easier. By the

```
trial {
   trial_type = first_response;
   trial_duration = 5000;

   picture {
      text {caption = "Press correct key"; };
      x = 0; y = 0;
   };
   time = 0;

   picture {
      bitmap {filename = "pic.jpg"; } pic;
      x = -90; y = 100;
   } pic2;
   time = 2000;
   code = "pic2";
   target_button = 1;
} example_trial;
```

Listing 4.1: SDL example: Definition of a trial object containing two stimulus events. The trial lasts for five seconds or until a button is pressed. Immediately after starting the trial a picture containing a text is shown. After two seconds a graphic file is displayed and the event is reported in the log file (using the code parameter).

time of execution the files are already merged, anyway. The same is true for SDL variables and features like loops and if-conditions. They are not actual programming structures, but text replacements and are preprocessed before the scenario is run. SDL variables, for example, are replaced with their values as the scenario is read. They serve for convenience and are no run-time variables.

No programming structures but convenience functions in SDL

The preceding paragraph and already the recipe analogy made clear that SDL is not a programming language. Its statements are no run-time instructions and are processed previously to the scenario start. SDL just specifies the set of scenario objects that will be used during the scenario and produces a trial order list for the trial objects defined in SDL. If there is no PCL program, all trial objects will be automatically presented in accordance with the trial order list. Since a trial object contains a list of stimuli and associated timing parameters, presenting a trial means, that

SDL is not a programming language

Without PCL: Automatic display of trials in specific order

Presentation displays the stimulus sequences contained in the trial. Although the automatic presentation of trials in a specific order has some feedback capabilities, in general it is impossible to implement response-dependent behavior in a scenario without using PCL.

PCL

PCL is a programming language

In contrast to SDL, PCL is a real programming language. It allows to provide a scenario with enhanced custom behavior and offers great flexibility in the design of experiments. In scenarios containing PCL, Presentation does not automatically display the trials as determined in the trial order list in SDL. Instead, the PCL program is executed and controls the scenario. It is compiled into an intermediate form and then the instructions are interpreted at run-time.

PCL has access to run-time information and scenario objects

PCL has access to all scenario objects described in SDL and can manipulate them. Additionally, the PCL program has access to run-time information provided by Presentation, such as button presses, classifications of responses, or temporal data.

PCL supports response-dependent experiments

In PCL real programming structures exist, e. g. loops, if-statements, arrays, and run-time variables (e. g. the basic types int, bool, double, string). PCL provides tools for incorporating randomized stimulus variation and offers tools, such as subroutines, that allow efficient programming. Thus, in contrast to SDL, it is possible to define and to call methods. Many predefined parameterized methods of Presentation for all different kinds of objects can also be used. Furthermore, PCL can support virtually any response-dependent experimental paradigm. An example for PCL can be seen in listing 4.2.

Criticisms

The use of two languages may have the advantage that specifications and instructions are clearer separated. One language is for the easy part, one for the complex part and

```
begin_pcl;

loop
        int i = 1
until
        i > trials.count()
begin
        trials[i].present();
        i = i + 1
end;
```

Listing 4.2: PCL example: Program that turns a non-PCL scenario into one that uses PCL to do exactly the same thing. The trials are presented in accordance with SDL's trial order list, but explicitly and not automatically.

the complex part is not needed in every case. But two languages have also clear disadvantages.

PCL requires complete redefinition of scenario structure

First of all, as soon as PCL is used, the whole scenario structure has to be defined again. If a user has almost finished an experiment and discovers then that he needs some PCL functionality, all ordering of trials in SDL is meaningless. Instead, the PCL program has to remodel the previous structure. It would be much easier, if SDL and PCL can also be mixed in some other way. Now, users have the problem that SDL meets its limits relatively fast, while PCL forms big obstacles for the reorganization of their existing work.

PCL has structural shortcomings

PCL has also some structural shortcomings. It provides many functions that can be called for writing into scenario objects to modify them, but there are only very limited possibilities for reading scenario object contents. For example, it is not possible to read out the name of a bitmap that is contained in a picture object. This can make simple things very complicated.

The Contextual Inquiry, described in the next section, was also conducted to see the effects of these deficiencies.

4.1.2 Future Developments

Graphical interface is planned

Finally, it should be mentioned that Neurobehavioral Systems has recently announced that they are currently developing some new features for Presentation. Their goal is to improve the ease-of-use of Presentation. Therefore, they plan to release a graphical interface for experiment creation within the next year. This tool should allow to graphically edit stimulus objects, trials, and their associated parameters.

4.2 Contextual Inquiry

Users have problems with Presentation

Presentation is a powerful application and has capabilities to fully support the implementation of psychological experiments. However, users stated that working with Presentation is hard and that they meet many problems in its operation. As the designer of the new interface for Presentation, I cannot regard myself as a typical user of this system, though. I have an entire different technical background and I am used to programming. The typical psychologist, in contrast, has no or only little programming knowledge. Thus, I am not able to identify the problems that the real users of Presentation encounter in their daily work solely by myself.

Designer is not a typical user

Contextual Inquiry includes interviewing and observing users in their normal context

Therefore, there is a clear need to investigate current work practices with Presentation. For this initial user study I conduct a Contextual Inquiry which includes interviewing and observing users in their normal work context. With this technique I want to learn how the actual users work with Presentation and what they do with the system. Further I want to find out what problems they encounter, whether the system hinders their work, and how they try to overcome their difficulties. I expect from this study to understand the users' work and needs—so that I can design an interface on the basis of a deep understanding of the application domain.

4.2.1 Method

User-centered design method

Contextual Inquiry is a user-centered design method, part of the contextual design methodology introduced by Beyer and Holtzblatt [1998]. They describe their idea as follows: "The core premise of Contextual Inquiry is very simple: go where the customer works, observe the customer as he or she works, and talk to the customer about the work. Do that, and you can't help but gain a better understanding of your customer." Contextual Inquiry "is a field data-gathering technique that studies a few carefully selected individuals in depth to arrive at a fuller understanding of the work practice across all customers." The designer has to attend the users, because only they know their work practice.

Field data-gathering technique to understand work practice

Perform a contextual interview

A Contextual Inquiry is usually performed using a contextual interview: a one-on-one interaction in which the user does his own work and discusses it with the interviewer. The interview should be built on natural human ways of interacting. Giving the interviewer a list of rules, that says which things he should do, does not work well as he has to concentrate so much on following the rules that he cannot concentrate on the customer. It is more natural to act out of a simple, familiar model of relationship. Furthermore, keeping with a relationship model is much easier for the participants than following rules.

Natural to act out of simple relationship model

Master/apprentice model chosen for Contextual Inquiry

For Contextual Inquiry a relationship between master craftsman and apprentice is chosen as an effective model for collecting detailed data. This model naturally creates appropriate behaviors for both participants. Just as an apprentice learns skills from a master, the designer wants to learn about the users' work from its users and thus he acts as the apprentice. The master craftsman teaches by doing the work and explaining it while working. The apprentice learns by observing the master's activities.

User is the expert in relationship model

It must be clear that in this model the user is the expert, not the designer. The designer is not there to help with problems or answer questions, though he might be able to. The only exception to this rule is, if the user is so stuck that he will not be able to do any more of the work the designer wants to observe.

Structure and details of work are revealed

Teaching in the context of doing work has the advantage that the master does not have to think in advance about the structure of the work or how to present it. The implicit work structure becomes apparent while working and talking about the work, since both participants pay attention to it. Learning in the context of ongoing work is effective for the apprentice, because he can ask questions at anytime in the process. Additionally, seeing the work reveals its details and its structure and talking about the work while performing it, prevents generalization as all details are right there. In this way, even those actions and motivations become apparent, that the master is not aware of.

Master/apprentice model extended by four principles

The designer learns about the users' work in order to support it with technology. The users can shape the designer's understanding of how to achieve that right from the beginning. To learn about the users' work, the designer cannot afford the time an apprentice would spend and often he has to study not only one user's work but a widely varying work practice of many users in many different projects. Therefore, the master/apprentice model is only a starting point. To conduct successful interviews under these circumstances, an adaptation of the technique is necessary. For this, Beyer and Holtzblatt define four principles:

Go to the users' workplace

- *Context*: Go to the users' workplace and watch them doing their own work. Avoid to disturb the workflow to stay in the users' context. In this way, details and structure of the work can be exposed.

Talk and discuss with the users about their work

- *Partnership*: Talk to the users about their work and engage them in uncovering unarticulated aspects of work. Ask the users to explain certain actions and discuss the structure of work.

Develop a shared understanding with the user

- *Interpretation*: Interpret findings immediately. Develop a shared understanding with the user about the aspects of work that are important by discussing the interpretations.

Direct the inquiry with a clear focus

- *Focus*: Direct the inquiry from a clear understanding of your own purpose. That means, focus on the aspects of work that are relevant to the design.

4.2.2 Participants

For my Contextual Inquiry I observed five users on their workplace. Three of them are psychologists: Two are PhD students/research assistants; one studies psychology and philosophy and works as student assistant. Among the other two users is one computer mathematics student and one computer science student. Both work as student assistants. Two of the five participants are female; ages range from 22 to 33. All participants are experienced computer users, three have programming experiences apart from Presentation.

Novices and advanced Presentation users

Three of the users are Presentation novices learning Presentation just for a couple of weeks. The other two users have used Presentation for several years. They regard themselves not as experts, though, since they have still deficiencies in the programming of experiments. Especially PCL causes problems.

The participants were recruited from the Department of Psychiatry and Psychotherapy and from the Interdisciplinary Center for Clinical Research (IZKF) at University Hospital Aachen. They were selected according to the following criteria. The users should have reached different levels of Presentation expertise. Beginners should be interviewed as well as advanced users. In addition, I wanted the users to come from different disciplines to see whether this makes differences in the use of Presentation. Finally, it was required that the participants are currently programming experiments, i. e. they are actively working with Presentation during the inquiry.

4.2.3 Set-Up

The interviews lasted between three and four hours. They started with explaining the method, in particular the master/apprentice model and that the users take the role of the expert. Further, I told the users the goals of my study and explained what is important for me. In one study, two users participated together. This does not conform exactly to the method, as it is actually based on one-on-one interactions. Nevertheless it seemed necessary to do so, because it was their natural context to work together with Presentation. The remaining three interviews were conducted with one participant each.

Avoided to help the subjects

I have familiarized myself with Presentation before conducting the interviews to be able to better understand the findings. Except for a few situations, I did not, however, assist the participants during the interview. The only exceptions were, when the users were so stuck in their work that they would not have been able to continue in reasonable time. In such situations I gave some small hints, mostly in interrogative form, to enable them to find a solution. In other cases, when the participants repeatedly applied unnecessary laborious strategies, I inquired whether there are better ways of doing that action. I hoped, in this way, the users would question their habitual work structures and may be able to find more sophisticated strategies.

Participants were encouraged to think aloud

To observe the natural context, the normal workflow had to be kept. Therefore, I paid attention to avoid interfering with the users' work throughout the entire interview. I encouraged them to think-aloud while doing their work and to talk freely about the actions they were performing. I tried to understand the users' actions and their motivation. Occasionally, this demanded inquiring about particular actions and strategies and the reasons for them.

Participants expressed own ideas for improvement

At the end of the observation the findings were summarized and briefly discussed with the users to verify their correctness. Then I asked the users what improvements they would expect to their work from an improved interface. Finally, they were given the opportunity to state their own ideas how the system should be changed.

After the inquiry I offered help

After the inquiry, I was available to the users to answer questions about problems that have occurred in the course of the observation or about Presentation in general. In addition, I gave some hints and advices to help the users with their work.

4.2.4 Findings

This section presents the findings of the Contextual Inquiry. First, findings about Presentation and its operation in general are described. This is followed by user comments about more specific properties of Presentation. Next are findings concerning decision and time structures, which are of great importance for my work. Afterwards, I will present the users' expectations for an improved system and their own ideas for changing the system.

General Observations

- Presentation has excellent functionalities for doing psychological experiments. But the Presentation UI, consisting of SDL and PCL, causes hard operation.

  Presentation has excellent functionalities

- Many functions are not used. Even an advanced user estimated that he uses less than half of the possible functions, because he is frustrated and not motivated to familiarize himself with unknown functions.

  Many functions are not used

- Some activities are done unnecessarily manually. Users rather accept manual work instead of trying to figure out the elegant way of doing it, which they believe exists. For example, because the stimulus variation function is possibly complex, a manual pseudo randomization is performed. Even paper and pen were used in this process that soon led to little errors. Instead of trying to find a PCL function, the user rather neglects some aspects of the randomization.

  Users do much manual work, probably unnecessarily

- Users showed anxious behavior when changing something in the code, because afterwards problems without obvious reasons often occurred which led to tedious error searches. They complain, that for this reason changes to scripts cannot be done quickly.

  Users were anxious when changing code

- Users complain about laborious debugging. Sometimes long manual trial-and-error phases are necessary. It takes long time to find little syntax errors or typos in template tables.

  Laborious debugging

- Users are discontent with specifying stimuli as each

  Users are discontent with specifying stimuli has to be defined individually with all its parameters: Thousands of almost identical lines of code are generated which takes a long time. Even though this results in a repetitive structure with only minor differences, shortening is hard and users did not find a comfortable approach. Thus, scenario files soon become unclear and long.

Long, unclear files

Copy/paste errors

- Users are apt to make errors because they use copy and paste abundantly when defining stimuli.

Users make rare use of variables

- Users make only rare use of SDL variables. Thus: They fill out large template tables line for line with a couple of different values without considering an eventual later value change which would cause editing every single line. Using variables instead would make the filling in and changing faster and more comfortable.

Negative emotions arise

- Users sometimes feel "harassed" by Presentation, because it does not do what they want it to do.

Users have syntax problems

- Users have problems with the syntax. They do not know for sure when to set curly braces and semicolons or what the correct sequence of symbols and statements is.

Problems with scenario structure and programming

- One novice user has great difficulties with understanding Presentation's scenario structure. Functions and characteristics of certain scenario objects are not clear. Further, the time management of stimulus events is not understood. He has not realized that always explicit time structures have to be defined. Moreover, common programming structures are unfamiliar; for example, the concepts of defining and using variables and of naming objects for reuse. Neither the sense of variables nor their advantages are realized.

User Comments

SDL is quite simple and can be learned quickly

- SDL is a quite simple scripting language. It can be learned quickly: It is possible to have successes in a short time with trial and error.

- Templates are very useful, because not everything has to be written out in full.

Templates are useful

- They learned Presentation with the aid of the documentation and by looking at existing projects. Besides, the users' opinions about the documentation differ between being very helpful and being confusing because of missing details and missing explanations of programming concepts.

Users used documentation for learning

- Learning PCL is much more difficult. It is hard to understand its operation methods and the meaning of variables and instructions is partly not known.

Learning PCL is difficult

- Users avoid using PCL and even do not feel confident enough to learn it, although they regard it as more powerful, efficient, and comfortable. They expect the time of learning PCL to be too long and so they rather try to implement something in SDL, which can be insufficient and incomplete.

Users avoid PCL

- Instead of writing each script from scratch, most times existing files are adapted to the current project which can be become time-consuming, too.

Users adapt existing scripts

- Users still have to read a lot in the documentation for figuring out details and syntax.

Users consult the documentation a lot

- There are no problems with terms in the program: The common user language of the domain is used except for some computer science notions in the programming structures.

Presentation speaks users' language

- It can be time-consuming to extend standard experiments to be executable with fMRI. It is not known in advance whether scripts can be kept or have to be changed and whether everything will work as intended in the scanner.

Adaptation to fMRI problematic

Decision and Time Structures

Users avoid the creation of complex experiment structures because of the UI. Although it is possible and desirable to have experiments that can take varying courses dependent on the subject's responses, most of the experiments are

Users avoid complex experiment structures strictly linear or merely have capabilities for giving simple feedback. The reason for this is that linearity can be easily realized in SDL whereas response-dependence requires PCL.

Implementation of complex structures too difficult

Users do not know how complex decision structures and extended feedback could be implemented. They say that experiments are often not response-dependent, because the implementation would be too difficult. Thus, the design and planning of experiments is accommodated to fit Presentation! The users' work is restrained as experiments cannot be planned totally free. However, Presentation is absolutely necessary.

User Ideas and Expectations

Users want a GUI for Presentation

Finally, it was interesting to hear the users' own ideas about how the system should be changed. The users suggested a graphical interface for extending Presentation that would allow displaying time visually. They gave several ideas for how this could improve the system:

Easier stimulus creation

- Stimulus creation could be simplified. There could be dialogs for creating text stimuli directly with their parameters. Another dialog could be responsible for creating trials. It should have functions for importing texts and pictures and for choosing existing stimulus events.

WYSIWYG layouts for stimuli

- Experiments could be designed visually in "what you see is what you get" (WYSIWYG) manner like slides in Microsoft PowerPoint without having to use programming languages. The layout of stimuli could be directly designed. This would eliminate the need for executing a scenario in order to see whether a picture is on its intended position. Changes in the visual layout should cause changes in the SDL code and vice versa.

Direct color choosing

- Colors could be directly selected from a color palette instead of looking up and typing RGB values.

- A visual time line could support fMRI pulse control. The scanner pulses could be depicted in the time line and there could be functions for aligning stimuli to pulses and for specifying jitter times.

Visualization of fMRI

Users want to drag stimuli directly into the program

Furthermore, it was said that dragging and dropping of stimulus files into the program with automatic further processing, instead of specifying each one manually, would ease stimulus creation.

Users want a dynamic test mode

Users would appreciate to have a test mode which provides a time line for running back and forth through experiments at different speeds and that would allow to halt experiments. At the moment it is not possible to correct particular points directly and return to the experiment or to run experiments only partially. Instead the real-time test run has to be aborted and restarted in the beginning after detecting errors.

Users expect to save time and to become more motivated

Users expect from an improved experiment interface that the construction of scripts is faster so that they could save lots of time. In particular, they want to spend more of their time for reasonable activities and not for repetitive and manual tasks. They hope that realizing their ideas will be easier and that the experiments will better conform to their planning. Furthermore, they wish that experiment specifications become clearer so that it would be less confusing to edit experiments. These features would reduce their frustration and in consequence increase their motivation. In summary, they want to concentrate on the task instead of struggling with Presentation.

Results Discussion

A new UI is desirable

The results show that a new UI is desirable. There are issues that make designing and implementing experiments unnecessarily laborious and error-prone. Since the users are frustrated, they avoid trying unknown features and adjust their design to the shortcomings. Especially the avoidance of creating complex structures implies that Presentation restrains their work. Temporal decision structures are often not included, because the realization would be hard, complex, and expensive and they do not know how to implement it. Therefore, experiments are redesigned because of the hard design process in Presentation.

Differences between users with technical and psychological background

The two users with technical background had a completely different working manner than the users with psychological background. They benefit much from their programming experience and can transfer the knowledge in common programming languages to the Presentation languages. Thus, they were able to familiarize themselves more easily with Presentation. In a short training period they have acquired much more skills than the novice from psychology. They may be more proficient in dealing with PCL than the two users with long Presentation experience. They are willing to try the more complex, difficult constructs without being anxious or having reservations. Thereby they are directly looking for elegant and efficient solutions and are not afraid to use PCL—in contrast to the psychologists.

Needs of domain users are not met

Obviously, the needs of the users of this application domain are not met in Presentation. The target group of Presentation are psychologists and not computer scientists. But computer scientists have clear advantages over the domain experts in operating the UI and specifying experiments.

Chapter 5

First Prototype: Paper

"First, solve the problem. Then, write the code."

—John Johnson

Paper prototyping for early usability testing

Paper prototyping is a variation of usability testing for evaluating designs in an early stage of the design process. "Its purpose is to get quick feedback from users while the design is still (literally) 'on the drawing board.' " [Snyder, 2003].

Typical tasks have to be defined

Since paper prototypes usually cannot reflect all aspects and features of the intended system, the scope of the prototype has to be limited. Therefore, first typical example tasks, that users of the system are expected to perform, must be defined. The prototype is created by drawing rough, even hand-sketched, drafts of the interface on paper. All interface elements that are necessary for performing the tasks, such as windows and dialogs, must be included in the prototype. Then, the paper prototype is evaluated with representative users who interact directly with its paper interface. The users do not get explanations for how the interface is intended to work in advance. The designer "plays computer" and simulates the reactions of the application to the user input by manipulating the prototype.

Interface is sketched on paper

Representative users test the interface

Paper prototyping is cheap and fast

Paper prototypes can be implemented quickly with low costs and aim at obtaining fundamental, high-level feed-

Fundamental, high-level feedback

back. This makes them an excellent means for realizing and testing first designs. The lack of details and the rough, unfinished look have the effects that users concentrate on general aspects of the design and that they are not inhibited to express negative feedback. In this way, fundamental results about the understanding of the system's concept can be obtained and the interface elements that work well or that cause problems can be easily identified.

In order to find out if the concept of time-based decision trees is understood by the users, I chose paper prototyping for the first iteration of the iterative user-centered design process. Further, I wanted to test my initial design ideas for the Presentation Visual Editor before implementing an interactive prototype.

5.1 Design

Design bases on Presentation

The Presentation Visual Editor interface is designed on the basis of the two languages SDL and PCL. That means, that all functions base on Presentation's functions, and the structure of scenario objects in Presentation is reflected in this interface. This system is not intended to be an independent experimental design and control program—it is a graphical extension to Presentation. The final system is planned to be capable of exporting the experiments to Presentation, i. e. it will generate SDL and PCL code, that can be executed by Presentation to run the experiments.

Goal is to generate Presentation code

Only visual stimuli and no fMRI support

The design of this prototype concentrates solely on experiments containing visual stimuli and omits other stimulus types. This was motivated by the Contextual Inquiry, in which I learned that experiments consist predominately of visual stimuli. Furthermore, the interface does not provide functions for fMRI experiments.

No variables

I decided to construct the prototype without offering the possibility to use variables in order to keep the interface simple. This can be considered unproblematic, because the users make rare use of variables in Presentation anyway, and partly do not even understand its sense.

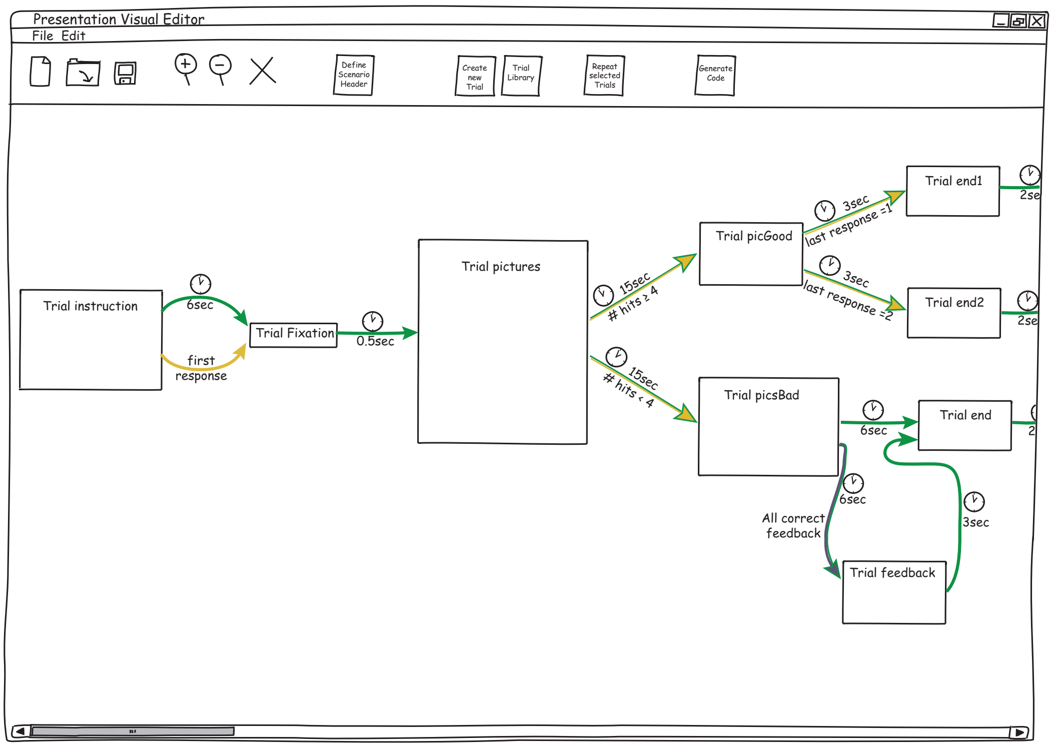


Figure 5.1: Main screen of the prototype showing an example experiment. The toolbar contains buttons for creating, loading, and saving scenarios. The magnifier icons zoom the tree structure and the cross deletes elements. Further, dialogs for editing the header and creating trials can be opened. *Trial Library* shows a library that allows to reuse existing trials in the scenario. The next button is responsible for repeating trial sequences. *Generate Code* converts the scenario to Presentation.

Throughout the design, I chose terms from the users' domain and from Presentation to denote functions and objects. This ensures understandability and enables an easy transition from Presentation. For example, no notions from graph theory are used, when referring to decision trees.

Notations from the domain and from Presentation

In the main screen, shown in figure 5.1, the experiment structure is constructed visually by using time-based decision trees. Scenario structures become less abstract as the temporal and logical arrangement of trials can be directly seen. A toolbar button opens a dialog for creating a new trial with its associated parameters. Trials are displayed in the main area and can be rearranged freely. Their visual size gives a hint for their duration.

Time-based decision trees used for visual experiment construction

Edges labeled with conditions and time constraints

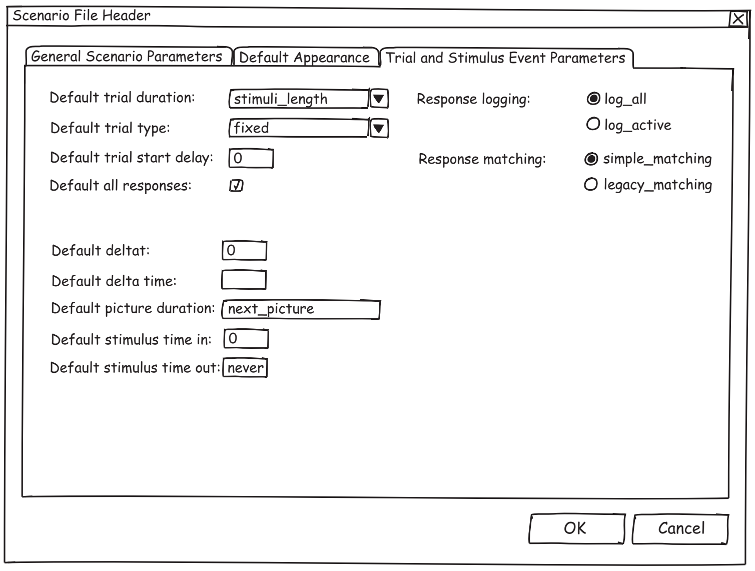
Trials are connected by directed edges which are labeled with time constraints and conditions. Transitions that depend solely on time constraints are displayed as green arrows whereas edges with response-dependent conditions are yellow. The prototype leaves open how these connections and their conditions are exactly defined.

Repeated structures visualized as one block

Repeating template structures can be realized by selecting a sequence of trials and forming a trial compound. This template structure is visualized as one big repeating block, instead of displaying all repetitions in a row, to keep the structure compact.

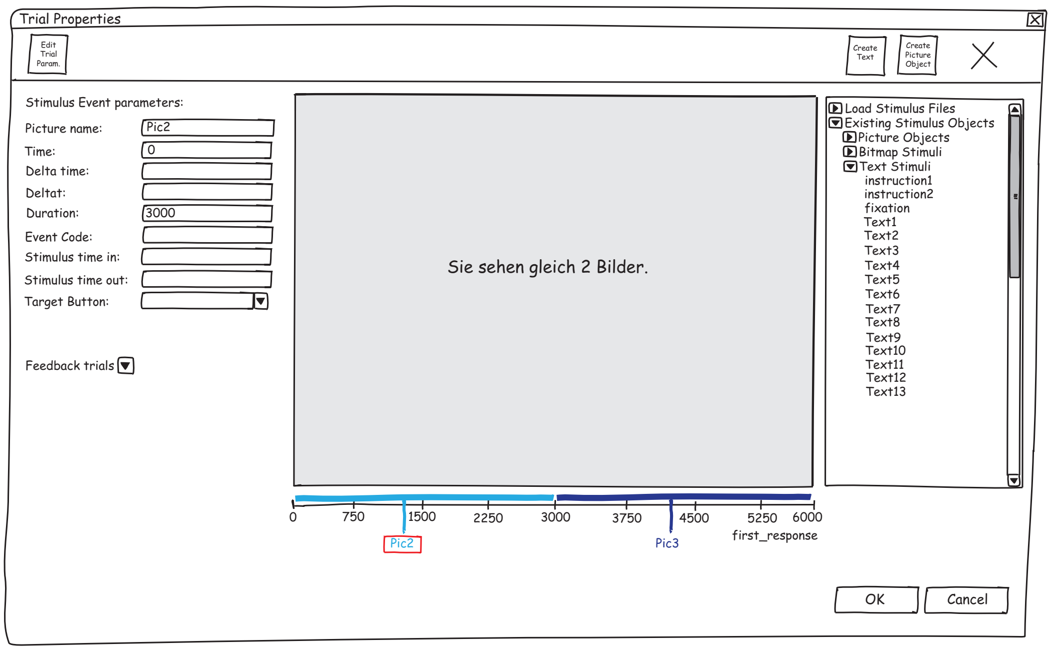
Dialog allows to specify the scenario header

Figure 5.2 shows the dialog for editing the scenario header. This dialog, divided into three tabs, allows to specify header parameters, such as the scenario name or the number of response buttons. Furthermore, default values for appearance, trial, and stimulus parameters can be defined, e. g. for font size or for the trial duration. In contrast to Presentation, users can directly see and select parameters and define their values. I limited the header parameters to a selection which is meaningful for the prototype. The parameters are preset with Presentation's default values.


Scenario File Header
General Scenario Parameters
Default Appearance
Trial and Stimulus Event Parameters
OK
Cancel

Default trial duration: stimuli_length
Default trial type: fixed
Default trial start delay: 0
Default all responses: ☑
Response logging: ● log_all ○ log_active
Response matching: ● simple_matching ○ legacy_matching
Default deltat: 0
Default delta time:
Default picture duration: next_picture
Default stimulus time in: 0
Default stimulus time out: never

Figure 5.2: Header dialog for specifying trial and stimulus event parameters. Other tabs can be found in appendix A.



Trial Properties

Stimulus Event parameters:

Picture name:	Pic2
Time:	0
Delta time:	
Deltat:	
Duration:	3000
Event Code:	
Stimulus time in:	
Stimulus time out:	
Target Button:	

Feedback trials

Sie sehen gleich 2 Bilder.

Figure 5.3: Trial properties window. This trial contains two consecutive stimulus events. The first one, *Pic2*, is currently displayed in the layout area.

The editing of trials is done in the trial window (figure 5.3). In its center area the layout of picture objects, which can contain several texts and bitmaps, can be designed in WYSIWYG manner. The stimulus parts can directly be manipulated, i. e. sizes and positions can be adjusted. Buttons are used for creating new picture objects and text stimuli. A library panel on the right side allows loading of stimulus files as well as reusing existing objects. Stimuli are inserted into the picture by drag and drop. In addition, stimulus files can be dragged directly from the Windows Explorer into this screen.

Direct manipulation of stimulus layouts

On the left-hand side, event parameters, associated with the current picture stimulus, can be specified. Beneath the layout area is a time line, displaying the total trial duration and the sequence of stimulus events. Their length can be changed just by moving the endpoints on the time line—or by typing values in the appropriate parameter field. Users can switch to a specific stimulus event by selecting its segment on the time line.

Time line visualizes the course of events and allows direct manipulation of times

Dialog for further picture properties

Double-clicking the layout area opens a picture properties dialog, which offers further picture editing functions. Besides functions, that are already contained in the trial window, like creating text stimuli, it is possible to define the background color and to see exact coordinates of the current picture part positions. As it was suggested during the Contextual Inquiry in chapter 4.2.4, colors can be chosen from a color palette that offers 18 predefined colors. Users do not have to type RGB-values anymore.

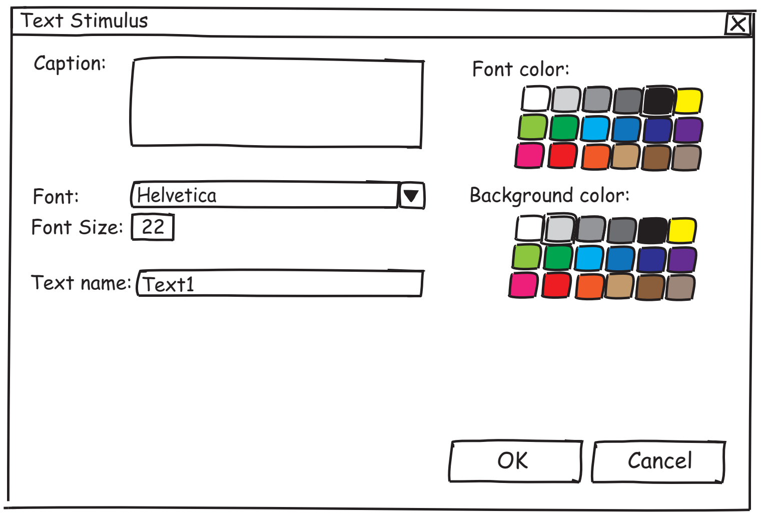
Direct color choosing

Dialog for precise scaling of bitmaps

Double-clicking a bitmap object leads to a dialog for setting its parameters. The window displays the current image and allows naming the bitmap. In addition, it is possible to scale the image. This is useful, if the scaling via direct manipulation is too imprecise. The size can be changed using a fixed aspect ratio. Users can specify exact values for height or width or use a scale factor. They can also set arbitrary values for stretching the image.

Dialog for text stimulus creation

The text stimulus window, shown in figure 5.4, is responsible for creating and modifying texts. Users can set a text's content, its name, and their favored font and its size. The text's font and background colors can be selected from a color palette.


Text Stimulus

Caption:

Font: Helvetica

Font Size: 22

Text name: Text1

Font color:

Background color:

Figure 5.4: Text stimulus dialog for editing text stimuli.

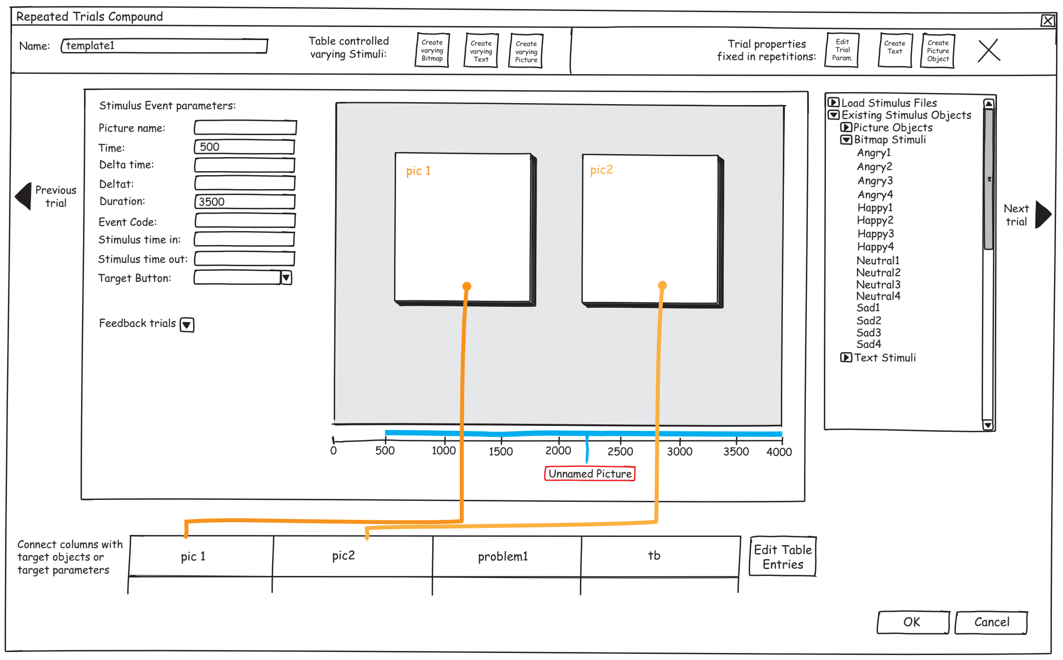


Figure 5.5: The window for editing repeated trial compounds. All bitmap stimuli contained in the *pic1* column of the template table are consecutively shown in the left varying bitmap field in the specified order; the ones in the *pic2* column in the right field.

Repeated trial compound window allows to vary stimulus presentation throughout the iterations

Finally, there is a window for editing the trials contained in repeated trial compounds (see figure 5.5). This window integrates the normal trial window and thus provides all of its functions, but it also includes functions for the template-like structure. Buttons for navigating to the previous and subsequent trial of the compound are located on its sides. In addition to standard stimulus objects, that will not change throughout the iterations, stimulus objects can be created, that will vary in each iteration. For example, there is button for adding varying bitmaps to the layout area. This results in displaying a bitmap placeholder graphic that can be manipulated just as normal bitmaps. Users have to drag all images, that they want to present on this position, into this field.

Stimulus order and certain combinations are crucial

In repeated structures, it is important to define the order in which the varying stimuli are shown. Furthermore, as soon as there are multiple varying stimuli, it is crucial to determine certain combinations of stimuli. This is achieved by using template tables that are similar to the ones that Presentation applies. The column names of such a table are displayed at the bottom of the window. These columns then have to be connected visually with the fields that contain varying elements by dragging lines between them. In the case of varying bitmaps, the names of all images, that have been dragged into one bitmap placeholder, will appear in the selected column. Columns can be connected with multiple fields. In particular, it is possible to connect them with stimulus event parameter fields. For example, if the bitmap column is additionally connected to the event code parameter, the log file will have an entry with the image that was shown in the iteration.

Visual specification of stimulus variation

The whole table can be viewed and modified through pressing a button next to it. Each line of the table corresponds to one iteration of the template. In this way, the stimulus order and combinations of stimuli are determined. It is possible to change the order by moving entries within a column or by moving entire lines. This dialog also provides a function for randomizing the lines of the table.

Some additional example screens from this paper prototype can be found in appendix A.

5.2 Implementation

Computer-aided prototype drawing

The interface was not directly hand-sketched on paper. Instead it was realized using a Wacom graphics tablet which allows drawing directly on its screen. The software Adobe Illustrator CS3 for Mac OS X was used for designing and editing the prototype. In comparison to hand painting this has the advantage, that modifications can be made easier. In addition, it is easy to show combinations of different layers. Each combination displays one state of the application. At the end, all interface elements were printed and cut out. Some windows were printed in multiple versions for showing different states.

5.3 Evaluation

5.3.1 Participants

I have evaluated the paper prototype with six users, who are all psychologists: four PhD students, one postdoctoral researcher, and one psychology student. Ages range from 25 to 35, and half of the participants are female.

The participants were recruited from the Department of Psychiatry and Psychotherapy and represent possible users. They cover a wide range of Presentation experience: A novice participated just as advanced users.

5.3.2 Set-Up

No explanation of the system

In the beginning of the study, the users were told that they will get a paper prototype of an alternative Presentation interface. I only told them that it is a graphical interface which allows building experiments visually. Further, I briefly explained the basic idea of paper prototyping. I encouraged the participants to think aloud while performing their tasks, and I asked them for each window to say what they think they can do and what the meaning of the elements and functions is. I emphasized that they should not hesitate to tell me about functions and notations that are not clear to them. This allowed me to find out when users have problems and do not know what to do or how to do it.

Subjects were encouraged to think aloud

Brief description of visual construction was necessary

Since the concept of using time-based decision trees for interaction is innovative and unfamiliar, it was necessary to explain shortly that trials have to be connected by conditional arrows, which determine the course of the experiment, as soon as two trials were created.

The test started for each subject on the empty main screen. All interface elements, that would be needed during the study, were prepared in advance. After a user had accomplished his tasks, a few retrospective questions about his impressions were asked. In total, a test took up to an hour.

5.3.3 Tasks

Three typical tasks should be accomplished

Each participant was asked to fulfill three tasks. They represent typical tasks, that users would perform for their real work, and are inspired by actual experiments that could be observed during the Contextual Inquiry.

1. *Linear experiment:* Create a simple experiment that starts with an instruction trial. The trial should contain two consecutive screens which greet the subject and explain him that he has to press one of two keys when two pictures are shown. The trial should last for six seconds or until a key is pressed. Then create a trial, that shows a fixation cross for half a second. Next is a trial, that should present the subject two images of faces simultaneously for four seconds or until a key is pressed. Finally, a last trial should thank the subject for participation.

2. *Repeated structures:* Create a second experiment. The structure is quite similar to the first one. It should start with an instruction, followed by a fixation cross and two pictures. But prior to the end trial, an additional trial should request a rating for the pictures, e. g. about the emotion shown on the images. This trial should last until the subject has given a rating. Furthermore, the middle part of the scenario, consisting of the fixation, the picture, and the rating trial, should be repeated several times before proceeding with the end trial. In each iteration, a different combination of pictures should be presented and the rating question should be adjusted accordingly.

3. *Complex non-linear experiment:* In the last experiment create a branching scenario structure. The experiment is response-dependent and changes its course in accordance to the subject's responses. (The scenario, that the users were asked to construct, is depicted in figure 5.1. The participants were not supposed to construct the interior trial structures, only the high-level experiment structure. I briefly described them possible events in the trials and the users' task was to create the branchings with the appropriate conditions.)

The scenarios in the tasks have an increasing complexity. They were chosen to cover nearly all features of the interface. The third task's superordinate target is to examine whether the users had fully understood the tree structure at the end of the study. Of course, the task serves also the purpose of testing the construction of complex structures and showing the users what can be achieved with the system.

Tasks were chosen to cover all features of the interface

5.4 Results

The observations and the direct user feedback during the evaluation of the paper prototype delivered very useful qualitative results about the design. These results are the basis for developing a refined prototype in the next iteration of the DIA cycle in chapter 6.

Evaluation delivered valuable qualitative results

All participants enjoyed working with the prototype and stated that the design is pleasant and provides meaningful improvements for designing experiments. They found the operation intuitive and understandable. After a short training period, the system would allow a fast and easy realization of experiments without being obliged to have much Presentation knowledge.

Interface improves designing experiments

Two users said, that the visual construction feels better than code writing. They emphasized that many of their colleagues have no programming knowledge and are not interested in acquiring some. For those people this system would be a great relief as no programming is needed.

Visual construction better than programming

Users said, that this interface is also easier than Presentation's, because the structure of experiments is visible. In comparison to SDL/PCL-code, the structure becomes more concrete. The participant's opinions on the time-based decision trees were consistently positive. They found the visualization clear and understood the concept quickly. Two users initially questioned the need for a branching tree structure as they have never done branching experiments before. But they recognized shortly the new possibilities and the potential for simple realization of adaptive experi-

Visible experiment structure was appreciated

Users quickly understood the concept of time-based decision trees ments, which will be of increasing importance in the future.

Visualization of repeated structures was liked

Combining repeated template structures into one visual block in the tree structure, instead of bloating the structure by showing all iterations in sequence, was regarded as a good idea. This visualization as one unit reflects the semantics of the experiment section well.

Many functions were immediately understood

The participants were able to use many features correct intuitively. They built a correct mental model of the system. They had no problems with entering the trial window and creating stimuli. They immediately wanted to drag and drop stimulus files into the program and into the layout area respectively and were pleased that this was possible. They enjoyed that they could directly manipulate the stimulus layout in WYSIWYG manner. All users instantly recognized the function of the time line in the trial window and manipulated it to change the timing parameters of stimulus events. It was stated that the time line makes time management less abstract as the durations and the order of stimulus events are visible.

Users liked visibility of parameter names and absence of syntax rules

Users generally liked about the graphical interface that they did not have to remember and type the names of all parameters. Specifying the values of parameters that they could see was very pleasing. This held true especially for the scenario header, because there are many different parameters that mostly are used only seldom. In this context, it was also considered as comfortable that they did not have to care about any syntax.

Omission of variables was approved

My decision to omit variables was approved by the test participants. They said, that variables would unnecessarily increase the complexity of the interface and confuse users.

Also usability issues

Besides these fundamental results, further results showed that there are also usability issues with the interface which should be improved in the next prototype. The encountered problems were discussed with the participants, which led to some valuable design suggestions:

Trial duration should be displayed inside the blocks

- The tree structure would be clearer, if the edges were not labeled with time constraints. Some users thought at first, the time would indicate a pause between the trials. Displaying the time inside a trial block seemed to be more intuitive for the users as the duration is a property of the trial. Only after the time has elapsed in a trial, the temporal transition is taken.

Some terms in the prototype are not intuitive

- Some terms used in the prototype are not intuitive. Users were for replacing *picture* (a whole screen of graphics) and *bitmap* (a single image) by *layout* and *image*, because the original Presentation notions can be easily confused. Though this might in turn confuse advanced Presentation users, they argued that the benefits for beginners would justify the change. For similar reasons, users also suggested to drop the trial duration value *forever* and the term *caption*, that denotes the content of a text stimulus. All in all, the chosen notations were clear, though.

Some buttons caused problems

- Some buttons should be revised. In the main screen, the icon for loading a scenario was interpreted once as closing symbol. Two users did not understand the meaning of the *Generate Code* button. In addition, the need for magnifying/demagnifying buttons was queried. Using the scroll wheel of the mouse instead, was considered more comfortable. In the *Repeated Trial Compound* window, users criticized the inconspicuous buttons for navigating through the trials and missed a button for adding further columns to the template table. In general, however, the users were content with the accessibility and comprehensibility of the buttons.

Name of a trial is not displayed

- Users pointed out, that I forgot to display the name of the current trial in the trial window.

Stimulus event parameters section confused users

- In the trial window (fig. 5.3), users were confused by the stimulus event parameters section on the left side. This section is always shown, even if no stimulus event has been defined, yet. In this case, the parameter fields falsely afford to be edited, and they attracted all users' attention when entering the window for the first time. Thus, if there is no stimulus event, the parameter fields should not be editable. Since it is not always necessary to use the parameters at all (e. g. timing can be specified using the time line), it might be meaningful to hide the fields by default and to show them only by request.

Picture properties dialog is distracting and redundant

- The dialog window for editing picture properties seemed to be distracting in the interaction and somehow redundant. Since it provides not much functionality, it might be a good idea to integrate those functions into other interface parts and drop this window. An image of the window can be found in the appendix (figure A.7).

Users demanded tool tips for parameter meanings

- Almost all users stated that tool tips, for instantly giving a brief description of the parameter meanings, would be of great value in the final system.

Library panels were considered useful, but can be improved

- The library panels, that serve for reusing existing objects and loading files, were accepted and considered as being useful. It was suggested to indicate which objects have already been used in the scenario and to introduce a function for importing libraries from other scenarios to allow faster stimulus creation.

Text alignment function is missing

- One user missed a function for defining text alignments for multiline text stimuli.

A function for managing multiple scenarios would be useful

- It was suggested to introduce a project function for managing multiple scenarios, since an experiment could possibly comprise several scenarios. This would eliminate the need for repeated closing and loading of scenarios, and it would combine several scenarios to a meaningful unit.

No user immediately understood the visual connection technique in the trial compound window

- Since the visual connection technique, chosen for creating template structures, is quite uncommon, no user did immediately understand the whole concept. Most users recognized quickly that they have to drag multiple images into fields for varying bitmaps to vary the stimulus presentation during the iterations. They also perceived the needs for establishing particular presentation orders and combinations of stimuli, but they did not figure out how do to it. They said, that it is necessary to see the visual connection function once to learn it, but then it is very understandable and easy to do. After overcoming the initial problems, they liked the concept. However, there could

After a demonstration the concept was liked be better visual cues for how this function works. Afterwards, they had no problems with handling the table, though they suggested to add more sophisticated randomization functions for it.

Users desired a test mode and fMRI visualization

- Just as in the Contextual Inquiry, users desired a test mode for controlling the experiment at different speeds. In this context, they also suggested to introduce a function for "disabling" trials, i. e. a function for choosing whether a trial will be translated to the Presentation code. This would allow to execute only certain sections of scenarios. Furthermore, they pointed out that it would be great to be able to design directly fMRI-capable scenarios and to have visualizations for fMRI features. I will treat these issues in chapter 9.

Chapter 6

Second Prototype: Paper

"The function of good software is to make the complex appear to be simple."

—Grady Booch

For the second iteration of the iterative user-centered design process, I decided to design another paper prototype that refines the first one. In this second prototype, I use the findings of the evaluation of the first prototype to modify the design. Further, I want to test some design ideas that came to my mind when observing the users. The goal is to simplify the prototype and to make its structure clearer.

Second, more refined, paper prototype

I decided against implementing a working interactive prototype at this point, because users are apt to focus on the details of functions and visual appearance when a system is already a working software application. But I want the users again to focus on the basic interactions and fundamental characteristics of the interface. From that I expect to obtain valuable high-level feedback about the changes in the system. I found this intermediate step necessary in order to be sure, how certain things should be realized in the software prototype. Furthermore, a paper prototype was chosen, because it allows testing the ideas faster and with less effort.

6.1 Design

Clarity of the layout was increased

The general layout of the interface has not been changed. However, several modifications of the design changed the look to some degree. I aimed at making it "cleaner" and tried to reduce the amount of information that is shown at once. My motivation was to improve the ease-of-use and to increase clarity.

Besides many analysis results of the previous DIA cycle, also some new design ideas are applied in the design of this prototype. Not all evaluation findings are incorporated, though. Some findings are postponed to the next design cycle, in which an interactive prototype is designed, others are discussed in the future work section in chapter 9.

Trial durations are shown inside the blocks

In this prototype, the layout of the time-based decision trees was modified. I added a start node as root for the trees to give them a clear beginning. Furthermore, trial durations are no longer displayed as time constraints of the edges. Instead the durations are shown inside the trial blocks.

Colors of the edges were changed

I also changed the colors of the edges to give them more meaning. Green edges are the standard edges, and they are taken, if the time has elapsed completely inside a trial. A green edge without any label thus means that this transition is taken automatically after the trial has ended on time and if no conditions of other edges are satisfied. An additional label means that the edge is taken, if the time has passed and the condition is satisfied. In contrast, red edges stand for premature endings of trials, for example, when a subject stops a trial by pressing a button. I chose the colors red and green, because their meanings of stop and proceed are well known in our culture. Thus, the new color coding may be understood at once.

Multiple scenarios can be managed

The system is now able to manage more than one scenario. Users have to define experiments, which can include multiple scenarios. The existing experiments and the contained scenarios are displayed in the main screen. Users can switch between different scenarios by clicking the desired scenario. The new main screen with the modified tree structure is shown in figure 6.1.

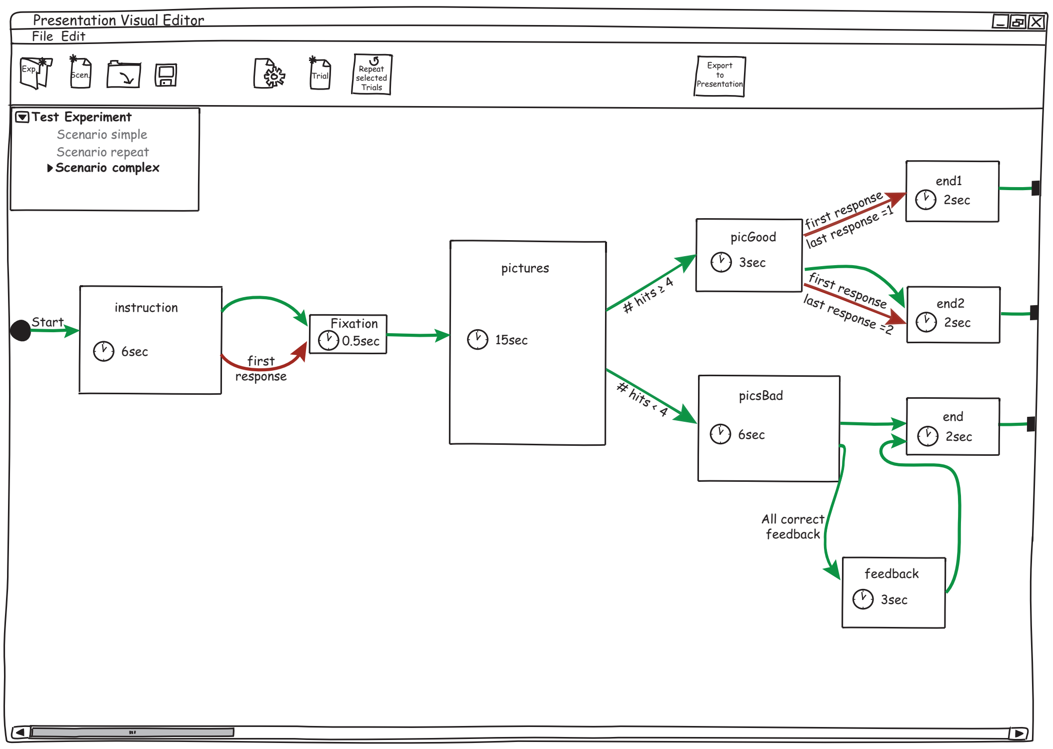


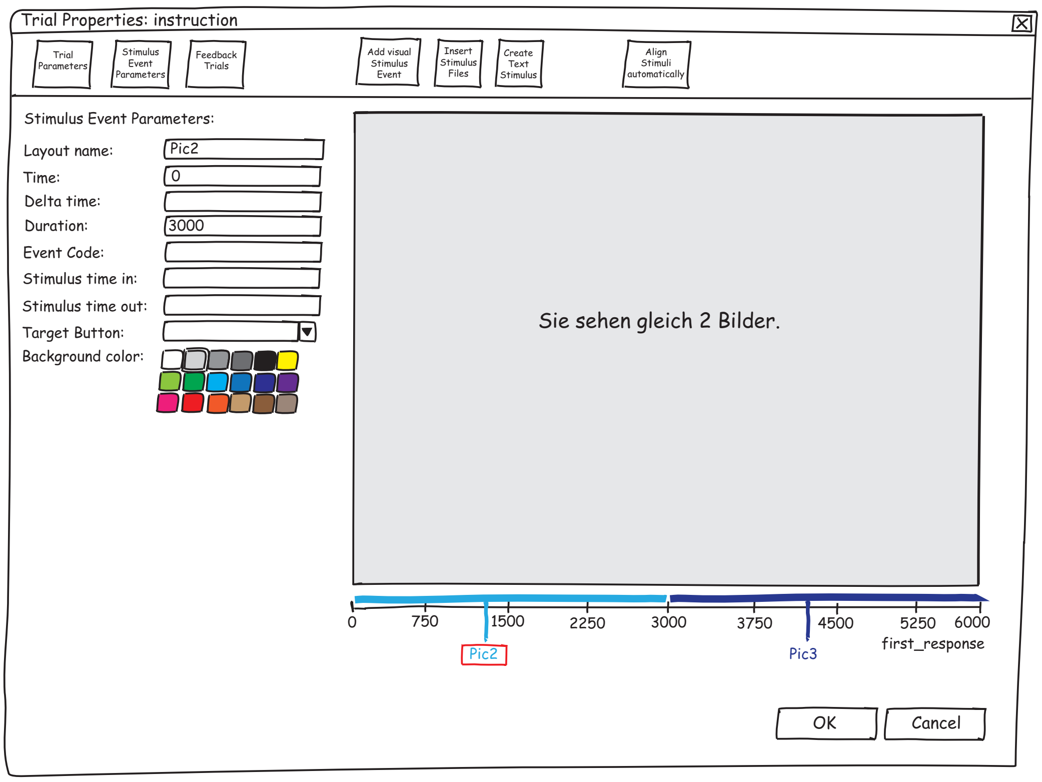
Figure 6.1: Main screen of the refined paper prototype showing the same experiment as in figure 5.1 in the last chapter.

Main screen toolbar was modified

In the main screen, the toolbar was revised, too. There are distinct buttons for creating experiments and scenarios. The buttons for loading and saving now refer to experiments instead of scenarios. Some of the purely text-based buttons were replaced or extended by icons. The buttons for zooming in and out were removed. It is planned to realize zooming with the mouse scroll wheel in the software prototype. The delete-button was also removed. Users should be able to delete tree elements using a context menu. Finally, the label of the button for generating Presentation code was changed to *Export to Presentation*.

More intuitive terms

Several notations in the design were exchanged. Motivated by the previous evaluation, I decided to use intuitive terms more consistently. The system calls visual stimuli now *layout* and *image* and talks no longer of *picture* and *bitmap* objects. The trial duration parameter value *forever* was replaced by *unlimited*. Further, the field for entering the content of text stimuli is simply denoted *Text* instead of *Caption*.



Stimulus Event Parameters:

Layout name:	Pic2
Time:	0
Delta time:	
Duration:	3000
Event Code:	
Stimulus time in:	
Stimulus time out:	
Target Button:	
Background color:	

Sie sehen gleich 2 Bilder.

Figure 6.2: Trial window from the second paper prototype with the same content as in figure 5.3 in the last chapter.

Trial name visible

The trial editing window, shown in figure 6.2, now displays the name of the trial. Its toolbar was also slightly reworked. Next to the button for editing the trial parameters is now a button for showing and hiding the stimulus event parameters section. As it was considered in chapter 5.4, the event parameters are hidden by default and can be shown, if needed.

Event parameters can be hidden

Feedback trials relocated

The parameters for specifying so-called feedback trials, that can be inserted after the current trial under specific conditions, were removed from this window. They were relocated in an extra dialog window, which can be called by a toolbar button, because it is a rarely used function.

Better button label

The button for creating picture objects was replaced with a button for adding visual stimulus events, since this reflects its function more precisely. It inserts a new event into the trial, preset with the values defined in the header dialog. An associated empty layout is automatically created with default settings. Further, a button for opening a standard file dialog was added that allows loading stimulus files into the layout. If the trial contains no stimulus events and a stimulus file is loaded or dragged from the Windows Explorer, a stimulus event and a layout are automatically created.

New file dialog button

The window, that was responsible for editing layouts, was removed from the design. Its functions were integrated into the trial window: A layout's background color can be chosen from a color palette in the event parameters section; the coordinates of selected stimulus parts are displayed; a button for aligning stimuli automatically was added to the toolbar.

Picture properties window removed

The window for managing repeated template structures was only slightly modified. The changes done in the normal trial window were adopted and the toolbar elements rearranged. Further, a button for adding a column to the template table was inserted into the window. In order to make them more conspicuously, the buttons for navigating through the trials were enlarged. The modified version of this window and an associated template table is shown in appendix A.

Repeated trial compound window slightly reworked

Finally, I decided to remove all libraries from the interface. That means that existing trials, layouts, images, and texts cannot be reused anymore. In my opinion, stimuli can be created quite easily in this system, so that the libraries are not essential. I did this to simplify and clear the interface. The windows now should appear tidier and less crowded. I made this decision, although the users were content with the libraries. Therefore, I am curious how the test users will react to this change.

All libraries were removed from the interface

6.2 Implementation

The implementation process of this prototype was identical to the first paper prototype (described in chapter 5.2).

6.3 Evaluation

6.3.1 Participants

The paper prototype was evaluated with six users, who are all psychologists: five PhD students and one psychology student. Ages range from 25 to 34, and four of the participants are female.

The participants were recruited from the Department of Psychiatry and Psychotherapy and represent potential users. They cover a wide range of Presentation experience: Two novices participated just as advanced users.

Four subjects tested first prototype before

Four of the participants have already taken part in the evaluation of the first prototype. I chose them to direct the focus of the feedback on the changes that had been made since the first version. The remaining two users, however, were unfamiliar with the system and its concept and were able to test it unbiased.

6.3.2 Set-Up

Second-time testers got no fixed tasks

The test participants, that were familiar with the first prototype, got a short reminder of the concept and were again encouraged to think aloud and to talk freely. These subjects were not asked to accomplish fixed tasks. They were supposed to explore the system on their own and to design some arbitrary experiment. Of course, they were constrained by the paper elements I had prepared. When it seemed advisable, I spontaneously gave them specific tasks to induce the use of certain functions. The users were asked to express their opinion on changes, if they noticed some. From time to time, I also pointed to some modifications.

Changes were discussed afterwards

When they used most of the functions and faced all changes, I ended the test run. Subsequently, I discussed the modifications with the users to hear whether they considered all changes an improvement. I also wanted to hear their overall impressions of how the system has evolved.

The two subjects, that had no experience with the Presentation Visual Editor before, were tested in the same way as the participants in the previous design cycle. The evaluation set-up was identical to the one described in chapter 5.3.2, and they were asked to perform the tasks from chapter 5.3.3. Only at the end, I deviated from that procedure and showed them additionally some windows from the first paper prototype. I found it interesting to hear how they evaluated the changes in the design.

First-time testers were treated like the subjects in the first evaluation

6.4 Results

The results from the first evaluation were confirmed in this study. Also the new test users quickly understood the concept and said that experiment design is much easier using the Presentation Visual Editor. The tree structure caused no problems; the connection arrows and their conditions were understood and applied correctly. At the end of the test, they were familiar with the time-based decision trees so that the complex tree of the last task could be constructed without any mentionable problems. They emphasized that the visibility of the structure was of great value.

Concept was understood quickly

Visibility of the structure is of great value

Similar to the first subjects, they understood many features of the interface immediately, e. g. they directly used the trial time line for specifying stimulus event durations. They enjoyed the direct manipulation of text and image stimuli in layouts and really liked the method to include stimulus files by drag and drop.

Direct manipulation and drag & drop were enjoyed

The two new users said that everything in the system can be intuitively understood without further explanation—except for the visual connections in repeated template structures. However, this was understood quickly as well after demonstrating it once. Then it was considered as a convenient and easy technique and they were able to apply it correctly. It was suggested to show the connections not all the time, because too many lines would make the screen confusing. A proposal therefor was to display the lines only when the mouse cursor is over a connected object.

Visual connections had to be demonstrated

Convenient, easy technique

Interface was simplified; seems clearer now

The modifications, that were incorporated into the design, were mostly approved. Only the removal of the libraries was disputed. The participants' general conclusion was that the interface was improved and simplified by the changes and that it seems clearer and more compact.

Tree layout was improved

It was said that displaying the trial durations inside the trial blocks is better than the first variant as the trees appear clearer now. This is also supported by the new color coding of the edges. The colors' temporal meaning is easily comprehensible.

Users were content with the new terms

The users were content with the new terms, I chose for some functions and objects. For instance, *unlimited* was regarded as a better parameter value than *forever* for a trial duration that is possibly unlimited (the trial can only be ended by a subject's response).

Removal of zoom buttons was accepted

The subjects accepted the removal of the zoom buttons in the main screen. In the paper prototype, however, it is not possible to zoom the trees at all. Thus, meaningful statements about zoom interactions cannot really be made at this stage.

Header icon caused problems

The icon that represents the scenario header was not identified as such by all participants. While it cannot be ruled out that this is due to the rough sketch, I will consider additional labels and tool tips for the buttons in the next prototype.

Multiple scenarios useful

Users found the new function for managing experiments with multiple scenarios useful.

Trial window seems easier and clearer

All participants stated that the trial window seems to be easier to operate and that it appears clearer, because less content is shown and the stimulus event parameters can be hidden. One subject said that "emptier, tidier windows with optional content are better".

Freely definable color palette is missing

While it was liked that colors for layouts can be directly chosen, one user missed a freely definable color palette. He said that it would be better to be able to select any possible color and to have the possibility to work additionally with RGB-values.

One participant missed a full-screen preview mode for layouts. He argued that this would allow to get a better impression of the stimulus presentation.

Full-screen preview should be added

No subject missed the removed window for editing layouts. They preferred to have its functions in the trial window.

Picture properties window not needed

In the window for repeated trial structures, all users found the navigation buttons better than before. In contrast to the first evaluation, the first-time users of the system had no problems to notice them.

Navigation in repeated trial structures easier

Two users said, that a trial library in the main screen is not necessary. In order to reuse trials, it would be simpler to copy and paste them. Two other subjects stated, that it could make sense to reintroduce it. It could ease the overview of large scenarios. They suggested to offer a function for showing it on request.

Trial library is not necessary

Opinions are less divided in the case of the stimulus libraries. All participants, who have known the first prototype before, argued for reintroducing it. Some wanted it back to the place where it was before, i. e. as regular window component, others were for optionally showing it on request or as some kind of own window. They argued that the reusing of objects can possibly accelerate the work. It is worth mentioning that the two new users, who saw the library not until the end of the test, said that the library is not necessary. But they as well, suggested that this feature could be included as optional content. However, that confirmed my opinion that it is not essential to include the library directly in the interactive prototype in the next design cycle. I definitely should consider it for a final system, though.

Users want stimulus libraries back

Libraries not essential

Finally, one user stated that it can be hard and time-consuming to understand unknown scenario structures in Presentation files and that the Presentation Visual Editor would really relieve such work.

Chapter 7

Third Prototype: Java

"There are two ways of constructing a software design. One way is to make it so simple that there are obviously no deficiencies. And the other way is to make it so complicated that there are no obvious deficiencies."

—C. A. R. Hoare

Working system for more detailed evaluation

In the design cycles so far, I have evaluated two paper prototypes and collected valuable high-level feedback about the basic interactions and fundamental characteristics of the system, which I could apply to improve my initial design. As next step, I want to evaluate my design and its interactions in more detail with a working system. Thus, for the third iteration of the user-centered design process, I implemented an interactive software prototype.

Software prototypes aim at low-level feedback

Software prototypes aim at low-level feedback about the details of the interface and its look and feel. They already look like a full application. However, they focus on the user interface and its interactions and are usually limited in their functionality. That means that not all functions, appearing in the interface, are necessarily implemented. Compared to paper prototypes, software prototypes look more polished and precise so that users get the impression that the interface is finished. Thus, they take the overall concept for granted and focus on the details of design and interactions.

Interactive prototype needed for testing the performance of the design

At this stage of the design process, a working prototype is also needed to be able to test the performance of the design. Besides qualitative results, an interactive prototype allows acquiring quantitative results about the quality of the interactions. After a pilot study, described in this chapter, I therefore conducted a further evaluation that compares the prototype with Presentation's original interface. This final evaluation is presented in chapter 8.

7.1 Design

Paper layout was adopted and polished

The visual appearance of the software prototype is very similar to the previous paper prototype. I adopted the layout and used more polished graphics for icons and the tree structure. In addition, some of the purely text-based buttons were enhanced with icons. Apart from that, I changed some elements of the design in accordance with the evaluation results of the last design cycle.

Functionality limited to most important features

I decided to limit the prototype's functionality to the most important features. Besides reasons of time, it was important to me to concentrate on the user interface and its interactions. The scope was to realize all interactions that are directly related to the experiment design, such as the tree structure or the stimulus layouts. In other words, I wanted to make everything functional that is necessary for proving the concept of the design. For this reason, I have not implemented the function for converting scenarios to Presentation code, yet.

Buttons have refined icons, labels, and tool tips

A screenshot of the main screen is shown in figure 7.1. All icons were refined to improve the look and to make the functions clearer. To ensure that all meanings are understood at once, all buttons contain a short text label and have a tool tip, that explains briefly the function. The button for converting the scenarios to Presentation now shows the logo of Presentation.

Load/save not functional

The load and save functions for experiments are not implemented in this prototype. Clicking the buttons only calls a file dialog without further functionality. Further, this proto-

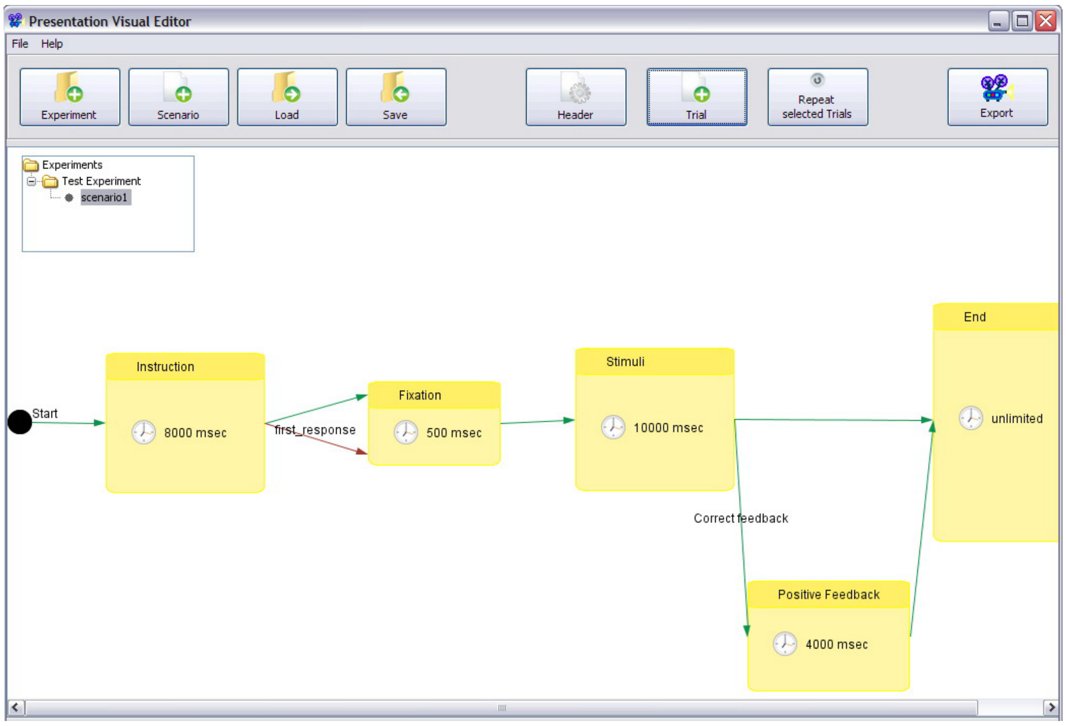


Figure 7.1: Main screen of the software prototype. There exists one experiment with one scenario included. This scenario is displayed as example experiment structure.

type can only handle one scenario. Users are able to add experiments and scenarios to the project overview panel and to rename them, but it is merely a mock-up. It is not possible to switch between different scenarios. Finally, the last function, that was ignored during the implementation, is the correspondent to Presentation's template files, the function for repeating selected trial sequences. I decided to omit this feature, because the effort of implementing it would have clearly exceeded its benefit in terms of gaining new insights.

Experiment projects and template structures not implemented

The general layout of the time-based decision trees was left unchanged. I added alignment guidelines that appear when trials are moved. This allows to align the trials with each other to get a nicer arrangement. Trials are connected by dragging an edge from one to the other. Each trial block has three anchor points for being connected on its left side: on the top, on the bottom, and in the middle.

General layout of time-based decision trees was left unchanged

Trials and connections now have context menus

I introduced context menus to the trials and to the connections. In this way, the trial window can be entered (in addition to double-clicking a trial), and it is possible to modify trial parameters, like the name or the duration, without having to enter the trial window. Further, the menu allows deleting a trial. The connection's context menu provides functions for deleting it and for calling the dialog window for configuring the associated conditions. The latter can be done by simply clicking the connection, too.

Connection conditions interaction now fully specified

The connection conditions window allows only the definition of conditions that are meaningful for the connected trials. Options for other conditions are hidden. If no response button is defined and the trial duration is fixed, only standard transitions without any label are allowed. If the trial can be prematurely ended by subject response, a corresponding box can be checked for this connection. If the user has defined response buttons in the header dialog, he can choose all kinds of response-dependent conditions. He does so by selecting condition types from a drop down list and specifying the desired value. A screenshot of such a window is displayed in figure 7.2.

Mouse wheel used for zooming

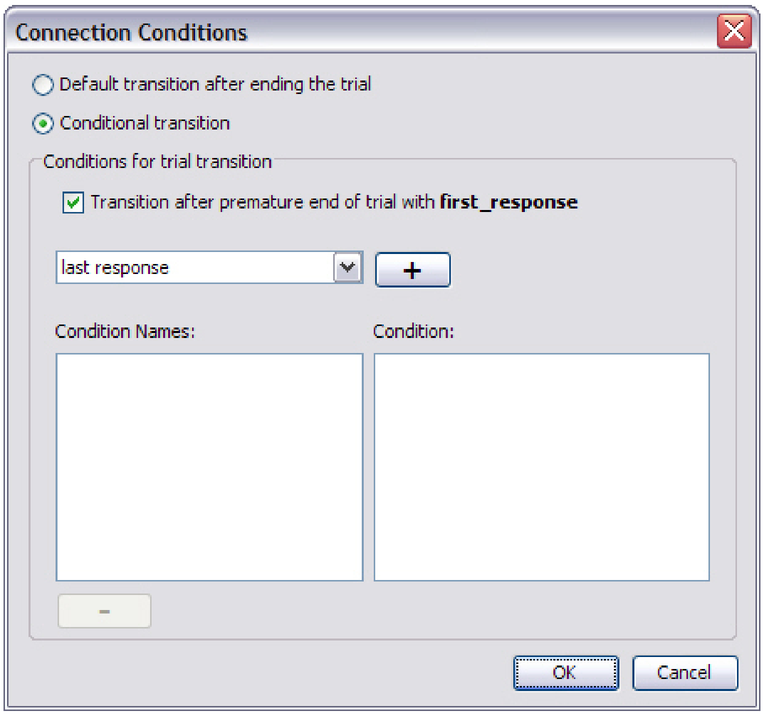
As it was planned, it is now possible to zoom the decision tree by scrolling the mouse wheel and pressing the control key. This is a common method for zooming in many other applications.

Trial window offers full-screen preview

The trial window is shown in figure 7.3. Now almost all of its buttons have icons. One button was added to the toolbar in response to user feedback in the last DIA cycle. It allows a full-screen preview of the currently displayed layout.

Time line is fully functional

The time line in this window is fully functional. Its scale is automatically adjusted to the trial duration. When it is used to change the duration or the start time of a stimulus event, the parameters on the left side of the window are adjusted appropriately, and vice versa. When these changes affect other events, the values and the visual representations are automatically updated for those events. This can happen, for example, when an event should last until the next one begins. The appearance of the time line segments was slightly modified. The name of the layout, that is associated with the event, is shown inside the segment.


Connection Conditions

( ) Default transition after ending the trial
(•) Conditional transition

Conditions for trial transition

[✓] Transition after premature end of trial with **first_response**

last response

Condition Names:

Condition:

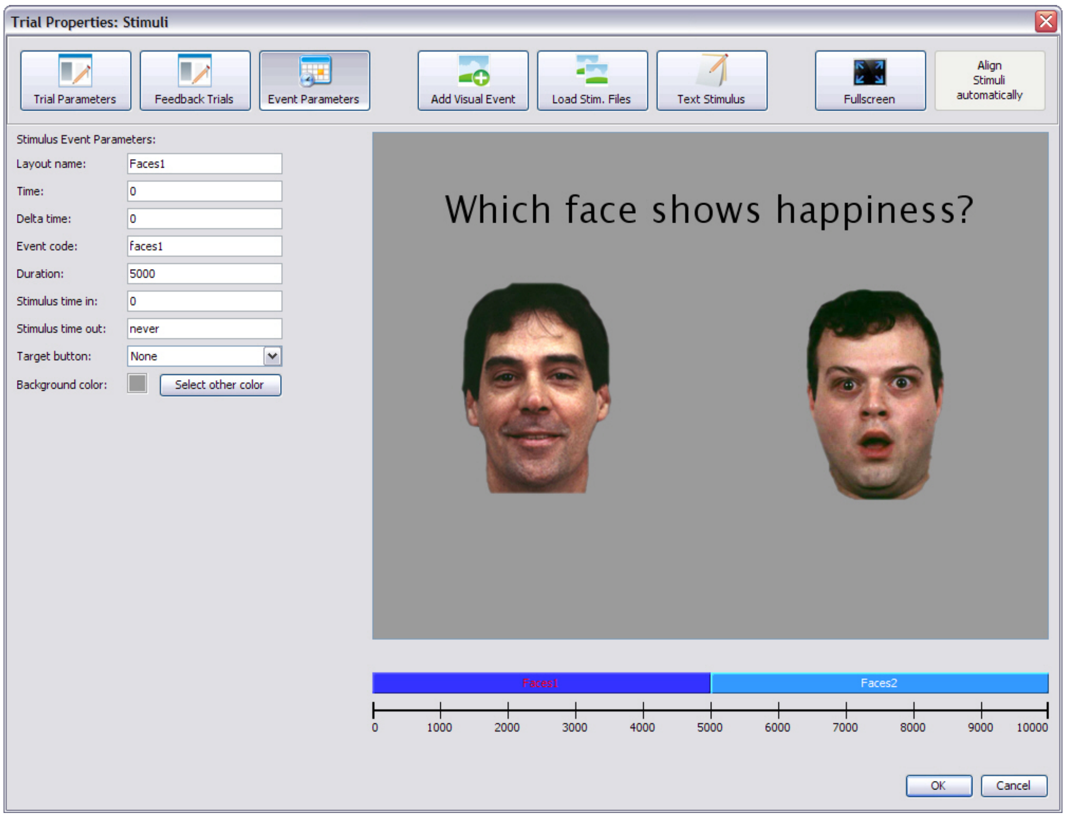
Figure 7.2: Screenshot of a connection conditions dialog showing all possible condition options.

After inserting image files into a layout, by dragging from the Windows Explorer or loading with the file dialog, they can be freely rearranged. Just as in the tree structure, alignment guidelines are shown. When the mouse cursor is over an image or text stimulus, corner points are shown, that allow resizing the object by dragging them. The stimuli can be removed by using a context menu. Double-clicking an object leads to the image properties and text properties dialog respectively.

Direct manipulation of stimulus layouts was realized

The image properties dialog, shown in figure 7.4, was extended by a function for restoring the original image size with one click, because such a standard function should be easily accessible. Therewith it is avoided that users have to click on the button for showing the scaling parameters and to set the scale factor to 1. When a picture is freely resized (without keeping the aspect ratio), a preview of the distortion is displayed directly in the window.

Image properties dialog shows preview of distorted image and can easily restore image



Stimulus Event Parameters:

Layout name:	Faces1
Time:	0
Delta time:	0
Event code:	faces1
Duration:	5000
Stimulus time in:	0
Stimulus time out:	never
Target button:	None
Background color:	Select other color

Figure 7.3: Screenshot of the trial window. The trial contains two consecutive stimulus events. The first event, *Faces1*, is currently displayed in the layout area. The layout presents two image stimuli and one text stimulus.

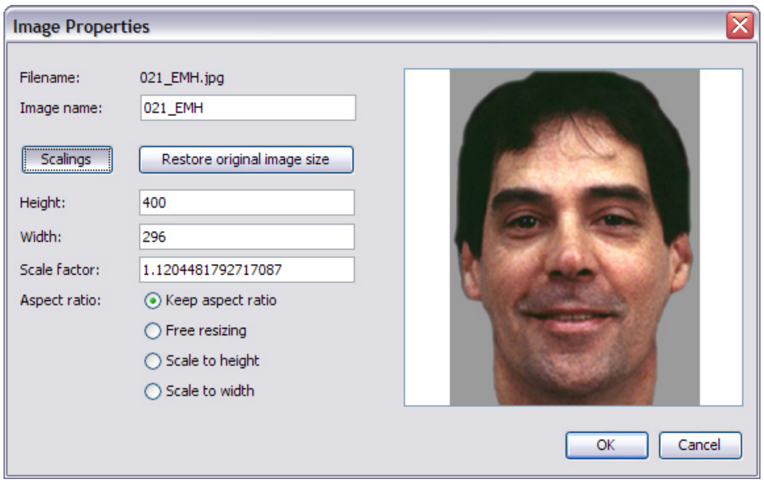
Text properties dialog offers text alignments

The text properties dialog stayed pretty much the same. However, I added a feature that was demanded in the evaluation of the first prototype in chapter 5.4. Now text alignments can be assigned to multiline texts. Hence, the header dialog was extended by default text alignments. A screenshot of the text stimulus dialog is presented in figure 7.5.

Users can freely choose colors

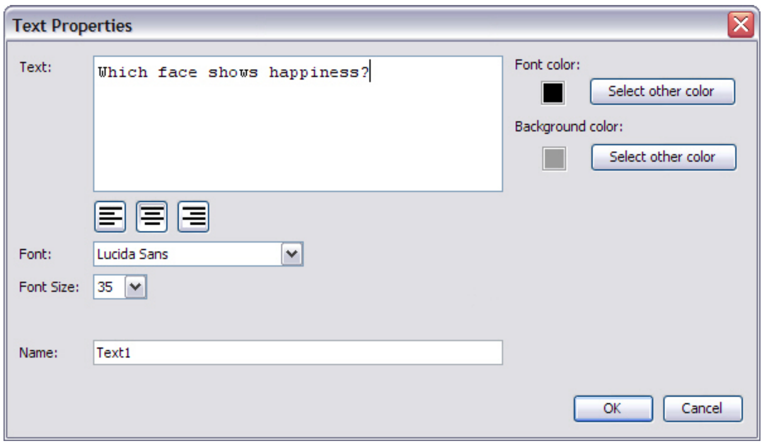
The method for choosing colors for layouts and text stimuli was improved. Users are no longer restricted to a predefined set of colors. Now a small field displays the currently selected color. Next to it is a button for calling a color chooser dialog. It provides support for choosing a predefined color from a palette, for choosing an arbitrary color from the color space, and for specifying RGB-values.

A sample run of this prototype is presented in appendix B.



Filename: 021_EMH.jpg
Image name: 021_EMH
Height: 400
Width: 296
Scale factor: 1.1204481792717087
Aspect ratio: [✓ Keep aspect ratio] Free resizing Scale to height Scale to width

Figure 7.4: Screenshot of the image properties dialog showing one of the image stimuli contained in the layout in figure 7.3 and its scalings.



Text: Which face shows happiness?
Font color: ■
Background color: ▨
Font: Lucida Sans
Font Size: 35
Name: Text1

Figure 7.5: Screenshot of the text properties dialog showing the text stimulus contained in the layout in figure 7.3.

7.2 Implementation

The interactive prototype was implemented entirely in Java (version 1.6.0). I chose this programming language, because I was already familiar with it and because it allows cross-platform compatibility. Although Presentation itself is limited to the Windows platform, there was no reason to

Software prototype was implemented in Java limit the prototype to one particular system, too. The prototype was developed and tested in Microsoft Windows XP and Windows Vista.

Prototype was developed with the NetBeans 6 IDE

In order to implement the prototype, I chose the software development tool NetBeans 6 developed by Sun Microsystems[1]. NetBeans is a free, open-source Integrated Development Environment (IDE) that supports Java and different other programming and scripting languages. NetBeans supports the development of full desktop, web, and mobile applications. An integrated Java Swing GUI builder allows an easy and quick creation of graphical user interfaces consisting of standard Windows components. Thus, NetBeans can also be used as rapid development environment, which makes it suitable for interactive software prototyping.

NetBeans suitable for interactive prototyping

7.2.1 Visualization

Visual Library API supports graph-oriented modeling

I preferred NetBeans to other Java IDEs, because it comes with the NetBeans Visual Library application programming interface (API). This is a powerful visualization API that supports graph-oriented modeling. It allows flexible implementations of graph structures and provides many methods of realizing dynamic behavior. This makes this API ideally suited for implementing the time-based decision tree view in the Presentation Visual Editor prototype.

Visual Library API realizes visualization and behavior of the tree structure

The tree visualization is realized by creating a Visual Library scene. The scene consists of three different layers that are responsible for displaying and managing the trials, the connections, and the actions assigned to the components. The API provides functions for adding actions to the whole scene and to the contained widgets. I used these actions, for example, to create hover effects, zooming capabilities, and the possibility of connecting trials by dragging. These actions made it also possible to call the trial window by double click and to show the context menus.

Tree elements are widgets

Trial blocks are implemented as Visual Library widgets. They are composed of several widgets to realize their dif-

[1]http://www.netbeans.org/

ferent parts. Connection widgets were used to create the edges in the tree. The API provides support for customizing the visual appearance and the behavior of the widgets.

Visual Library API was used for realizing the direct manipulation of stimuli

The Visual Library API was also applied to realize the layout area in the trial window. The layout area is also a scene and all displayed stimulus objects are realized with widgets. The action functions of the library were used to implement the direct manipulation of stimuli.

All visual representations of trials, edges, and stimuli are associated with data objects that determine what is shown.

7.2.2 Data Structure

Data structure is adapted to Presentation

The data structure of the program is adapted to Presentation's structure of scenario objects. This structure makes sense for the data modeling of experiments. A completely different structure would unnecessarily complicate the generation of SDL and PCL code. In short, that means that a scenario contains trials. Each trial includes stimulus events, and each stimulus event includes a layout. Finally, each layout can contain different types of stimulus objects.

PresentationVisualEditorView class represents a scenario

The *PresentationVisualEditorView* class creates the main screen of the interface and is in charge of creating the tree view and of painting the widgets. The class manages the set of trial objects (instances of the *Trial* class) and the set of connection objects (instances of the *Connection* class) that are contained in the experiment structure. Thus, this class represents a scenario. In order to extend the application to manage multiple scenarios, this part of the data structure has to be changed. In this case, a clearer distinction between the main screen view and the scenario data model is needed. It was sufficient for this prototype, though.

Trial objects store the set of contained stimulus events

Connection objects store their conditions

A *Trial* object stores all information that is necessary for a trial window and for painting the corresponding trial widget. In particular, it stores all the stimulus events (instances of the class *visStimEve*) that are presented in the trial. A *Connection* object contains the conditions that are associated with the corresponding edge.

VisStimEve objects store the contained layout and the therein contained stimulus objects

VisStimEve objects store the values of the associated stimulus event parameters. Further, each object stores a Visual Library scene that visualizes the contained layout object. These scenes are each shown in the layout area in the trial window, when the content of a stimulus event is displayed. All stimulus objects contained in a layout are also stored in an object of this class. That can be image stimuli (objects of the class *imageStim*) and text stimuli (objects of the class *textStim*).

imageStim objects store images; textStim objects store text stimuli

An object of the class *imageStim* includes an image that is used as stimulus. It is stored in two versions: in its current size (after possible resizing), and in its original form. The original image is stored to retain the best possible quality when the image is resized. Further, parameter values needed in the image properties dialog are stored. An object of the class *textStim* stores a text stimulus with all its properties.

7.3 Evaluation: Pilot Study

Pilot study is conducted to find unexpected problems

This evaluation of the software prototype is a pilot study. It is not supposed to evaluate the quality of the prototype exhaustively. Before conducting the full evaluation, that will also include a direct comparison to Presentation, I want to ensure that no unexpected problems in the operation of the prototype occur. For instance some interaction problems, that were hidden in the paper version could become evident. Users may perceive certain actions different, when they are performed in a real program instead of being simulated on paper. Since software prototypes afford low-level feedback and are more detailed, subjects may discover design issues that were not apparent or even nonexistent in the previous designs.

Pretest new interactions and detect bugs

This prototype also includes some new interactions, e. g. the definition of conditions for connections in the tree structure. Those functions should be pretested before doing the final study. Further, I want to ensure that the system does not contain any serious bugs.

Finally, I will use the impressions of the study to plan the final evaluation.

7.3.1 Participants

The software prototype was evaluated with three users, who are all psychologists: one PhD student, one postdoctoral researcher, and one psychology student. Ages range from 27 to 35, and all participants are male.

The participants were recruited from the Department of Psychiatry and Psychotherapy and represent potential users. All of them are experienced Presentation users.

7.3.2 Set-Up

The pilot study lasted about 40 minutes. The interactive prototype was installed on a notebook running Microsoft Windows XP, which was brought to the users' workplace.

No explanation of the system

The subjects were told that they will work with a software prototype for an alternative Presentation interface. I briefly explained to them, that it is a graphical interface that allows building experiments visually. I pointed out to them that it is a prototype, that can still have bugs, and that not every function they see is fully implemented.

Users were encouraged to think aloud

Similar to the previous evaluations, I encouraged them to talk freely about their impressions and to think aloud while testing. This allowed me to find out when users had problems and did not know what to do or how. I emphasized that they should not hesitate to tell me about functions that are not clear to them and that they can make suggestions for improvement at any time.

I avoided to explain them how the interface works. I made an exception for explaining the basic concept of using the time-based decision trees for experiment construction. Apart from that, I gave only small hints, e. g. that the control key has to be pressed for connecting trials or that stimulus parts, trials, and trial connections have context menus for further functionality.

Users explored the system on their own without fixed tasks

The subjects were not supposed to accomplish fixed tasks. They started the test on the empty main screen and were asked to explore the system on their own initiative. From time to time, I specified tasks that were tailored to the participant and the current situation. I also gave them some ideas for things that they could do. At the end, they ideally had used as many different functions and interactions as possible.

Afterwards, I discussed my observations and notes with the users, and I gave them a last opportunity to comment the prototype.

7.4 Results

Extreme positive feedback

The user feedback was extraordinarily positive. All participants said that there are no real problems in the operation of the system. One user explicitly praised its usability. They said that they like the Presentation Visual Editor and even expressed enthusiasm about it. Working with this system would increase their motivation when designing experiments.

Design is intuitive; system could be learned quickly

The participants stated about the general design that it is self-explanatory and intuitive and much clearer than Presentation. Although the interaction with time-based decision trees is an unfamiliar concept, which would cause great changes in the way of working, the initial training and the work with it would be easy.

Users liked the visibility of the experiment structure

The subjects commented about the time-based decision tree concept, that it is very good, that the entire experiment structure can be seen directly. Thus, coherences become clear more quickly.

System useful for unexperienced Presentation users

Two of the subjects emphasized, that the Presentation Visual Editor is particularly useful for unexperienced Presentation users, because one is able to build experiments fast without needing to know much about Presentation. The Presentation Visual Editor would provide "immediate functionality without long preliminary work."

Users were keen to experiment with the functions

One user told me that he has enjoyed trying the different functions; he liked especially the direct manipulation of stimuli. I also observed, that the users were keen to experiment with the functions and parameter values.

No problems with tree interaction

None of the participants had problems with understanding the interaction with the tree. They stated that the tree construction is convenient and intuitive. They quickly discovered the possibility to arrange the trials freely and liked it.

Direct manipulation understood quickly; problems in the implementation

The interaction with layouts and the creation and manipulation of stimuli were shortly understood. There are problems in the implementation of the direct manipulation, though. The direct adjustment of stimulus sizes by dragging their corners does not always work as intended. When it is done too fast or the "wrong" corner is used, errors occur.

Time line handling caused no problems; overlapping events are missing

The handling of the time line in the trial window caused no problems. One user criticized that it is not possible to realize overlapping, simultaneous stimulus events. This makes sense, when a stimulus event can be aborted by a subject's response. In this case, it should be possible to define another temporally overlapping stimulus event that will be shown as soon as the first one was ended.

No problems with specification of parameter values

The subjects had no problems with specifying parameter values. They used the intended input format for entry fields that require some kind of syntax, e. g. times in milliseconds or comma separated lists.

Zoom function useful but causes unnecessary navigations

Using the mouse scroll wheel to zoom the experiment structures was considered as good interaction. The participants said that the zoom function is very useful. When scenarios become big, it eases the overview. There was, however, one issue with its functionality. The zoom always focuses the same screen region so that users have to navigate a lot via the scroll bars to reach a certain area after having enlarged the view.

Copy function for trials is missing

The participants missed a function for copying trials. If they want to present a trial multiple times at different points in one scenario, they have to specify it each time again. It would be less laborious, if trials could simply be copied.

Chapter 8

Evaluation

"Enough research will tend to support your conclusions."

—Arthur Bloch

Comparative study to evaluate design in more detail

As last step of the iterative user-centered design process I conducted a final evaluation of the system. I wanted to test the interface and its usability in more detail to prove my design and its concept. This final evaluation goes beyond the previous user studies as the goal is not to capture mere qualitative but also quantitative results. Therefore, I conducted a comparative study that directly contrasts the Presentation Visual Editor with Presentation's original interface.

Measures the design performance

The subjects were asked to design an identical experiment with both systems. Therewith it was possible to measure the performance of the Presentation Visual Editor and its interactions. In addition, I prepared a questionnaire to collect the users' opinions about the system in a standardized form.

Find out whether time-based decision trees are beneficial

The goal of this evaluation is to find out whether the concept of time-based decision trees is understood and whether its application in interaction design is beneficial for the design and implementation of psychological experiments.

8.1 Changes to the Design

Before I started with the final evaluation, I made some changes to the design. These modifications concern specific functions and were motivated by the pilot study for the software prototype:

Debugged time line

- Some bugs in the implementation of the trial time line were fixed. Mutual influences of different stimulus event time conditions are realized more consistently.

Added overlapping stimulus events

- Overlapping stimulus events were introduced and are correctly visualized in the time line. There are two cases in which overlapping is possible: The event duration has either the value *response* or the value *target*. The former allows a subject to stop the event presentation by pressing any response button; for the latter a specific target button is needed. If such an event is interrupted, a possible overlapping event that was running in the background is presented. Overlapping makes no sense for other event timings, because those events have fixed durations and overlapping would never become apparent in an experiment.

Indication of trial start delays

- Start delays of trials are indicated more obviously next to the time line.

Better zoom function

- The zoom function was improved. Users can now determine which screen area is focused, when they are zooming in. There is always the area enlarged that the mouse cursor is currently pointing at.

Trials can be copied

- I introduced a copy function for trials to enable reuse. The context menu of trial blocks was extended by an appropriate entry that clones the respective trial.

8.2 Participants

Nine subjects participated in the final evaluation. Eight of them were psychologists: three postdoctoral researchers and five PhD students. One participant was a physician, but he also takes part in psychological research. Ages range from 24 to 36, and five of the participants are female. All participants were recruited from the Department of Psychiatry and Psychotherapy and represent potential users.

Experienced computer users

All of the subjects are experienced computer users working with computers for a minimum of ten years. Three of them had programming experience before using Presentation.

Varying Presentation experience

They all have Presentation experience to a varying degree. Their Presentation usage time ranges between a couple of weeks and five years with an average of 2.4 years. Their Presentation skills differ much. This was an important criterion when choosing the subjects. Except for one user, all have reworked existing scripts before. Five of them have implemented complete experiments on their own. But only three of them declared that they have used PCL in their work with Presentation. It is particularly noticeable that they generally assess their Presentation skills as poor. Five of them said that they are not capable of using Presentation to the full extent, two subjects were undecided, and only two users said that they are rather capable of using it completely.

Self-assessment of their skills is bad

Seven participants were completely unfamiliar with the system and the concept and did not participate in any of the previous evaluations. Two of the subjects had already knowledge of one of the paper prototypes as they participated in one evaluation two months before.

8.3 Set-Up

The final evaluation took between 60 and 90 minutes each. Similar to the pilot study, the interactive prototype was installed on a notebook running Microsoft Windows XP, which was brought to the users' workplace.

Short introduction

At the beginning of each study, I told the participant that I have developed an alternative interface for Presentation. I explained that it is a graphical extension that allows building experiments visually and that it is planned to create executable Presentation scripts. I emphasized that it is still a prototype that does not include all features a final system should have and briefly described the scope of my design. At this point, I gave no further explanations.

Experiment creation in both systems

I then outlined the structure of the study and told the subject that he is supposed to design an experiment with both the prototype and Presentation. While I prepared the system, the subject was asked to fill out the first three sections of a questionnaire.

Introductory task with the prototype

Since the subjects were all familiar with Presentation, but did not know my interface, I gave them a small introductory task that had to be accomplished with the prototype. In this way, the subjects got a first impression of the system and learned some of the interactions before the comparative study began. This approach enabled a more realistic and fair comparison.

Times were measured

After the introductory task, I explained the experiment that had to be realized and started the test. The time needed to perform the task was measured for both systems. Thereby, the test duration was limited to 30 minutes each. The order of the interfaces was varied throughout the subjects to neutralize learning effects.

Thinking aloud was encouraged

Similar to all previous evaluations, I encouraged the participants to talk freely about their impressions and to think aloud while testing. However, this was not required, because too much speaking could influence their task performance.

Explanations were avoided

I assisted the subjects as little as possible and avoided explanations. The same exceptions were made as described in chapter 7.3.2 for the pilot study. In addition, in some cases I explained the meanings of parameters to save the users from searching in the documentation. However, I paid attention to help them in equal measure in both systems.

Complex example tree had to be understood

After they had worked on the tasks, they were asked to complete the questionnaire. In between I showed them an example for a complex experiment structure in my interface to test their understanding of the time-based decision tree concept. They were supposed to explain the experiment

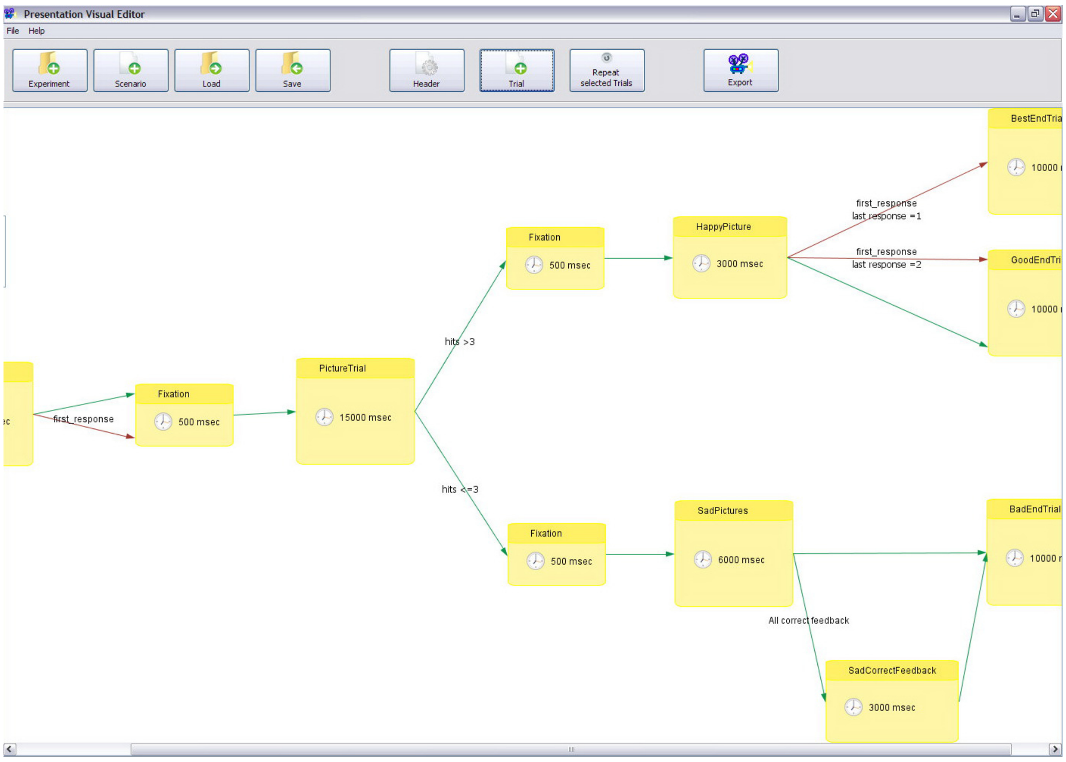


Figure 8.1: Example of a complex experiment structure that was shown to the users during the final evaluation.

to me. Additionally, some questions in the questionnaire referred to this example. A screenshot of the experiment structure can be seen in figure 8.1.

8.3.1 Questionnaire

Questionnaire for standardized and detailed results

To obtain standardized and more detailed results about the users' impressions and opinions, I prepared a questionnaire. The questionnaire is divided into ten sections. The first two collect profile data of the subject and information about their computer and Presentation experience. Then they were asked to assess their Presentation and computer skills, and I wanted to know their attitude to GUIs in the context of Presentation.

The fourth section inquires about their general impressions of the Presentation Visual Editor design and its performance. This is followed by questions about the time-based decision trees and the interaction with them. Then I asked them about the pros and cons of the GUI and of the combination of the Presentation Visual Editor and Presentation to realize experiments.

The next two sections are about the stimulus creation and manipulation in the prototype and about the trial time line. After that, I asked questions about their understanding of the complex tree example and about their confidence to build such an experiment on their own with my program and with Presentation.

While almost all of the preceding questions use a five-level Likert scale to obtain answers, the questionnaire concludes with five open questions, e. g. about missed functions and general comments. The entire questionnaire can be found in appendix C.

8.4 Tasks

Simple introductory task

The introductory task, the participants were asked to accomplish with the prototype, was quite simple. They had to create a trial that presents an image for a certain time. In addition, they had to determine a target button for the stimulus event.

Main task was mood induction experiment

The main experiment, that had to be realized with both systems, is a simple mood induction experiment. It starts with a trial for presenting an instruction. The instruction explains that the subject will see five pictures and that he has to rate his mood afterwards. The instruction can be aborted by subject response. After the instruction has ended, the next trial consecutively presents five pictures of happy faces. The trial duration is fixed and the images are shown two seconds each. In the following trial, the subject is asked how happy he is now. An image of a scale requests him to rate his happiness using the number keys 1–5. The rating input ends the trial. If the subject has declared that he is happy, that means he has pressed 4 or 5, he will see an end trial that appreciates his participation. Otherwise, the image trial and the rating trial are shown again.

Users were allowed to reuse existing script

In order to establish a realistic work situation, the participants were allowed to use an existing script from a real experiment for their work with Presentation. This scenario contains some structures that could be directly applied in their task. This is more realistic, because most often they do not start from scratch when writing scenarios but rather reuse existing scripts.

Task should be short, typical, not too hard

Several criteria were decisive for the design of the task. First of all, the experiment realization must be practicable within 30 minutes. Therefore, the experiment must be comparatively short and must not include too many different or difficult features. Further, the experiment should be a typical task that could be designed similarly for actual work. That means, the tasks within the experiment realization should be common to the users. In addition, the experiment creation must not be too hard. Every participant should have a real chance of succeeding in both systems. This is why the scenario is linear and simple for the most part as basically just a couple of text and bitmap stimuli have to be included into trials.

Some complexity

Finally, I decided to add some complexity to the scenario as there is one branching. In Presentation, this can only be achieved with PCL and I am aware that participants might fail for this reason. However, this branching is the last step in the implementation with Presentation and I will keep track of the time the participants need for the simple part of the experiment, too. In this way, I will be able to measure the performance of the prototype for simple, linear structures as well as for complex structures.

8.5 Results

Prototype clearly outperformed Presentation

The Presentation Visual Editor clearly outperformed Presentation in the realization of the task experiment. The participants needed on average 21 minutes (standard deviation 4 minutes) for accomplishing the task in the Presentation Visual Editor. In contrast, only one of the nine subjects

Users failed in Presentation

succeeded in creating the experiment in Presentation in the given 30 minutes. Two users failed completely, at most they created one of the trials. Four of the other subjects developed all trials and realized the simple linear structure. This took them on average 40 % of the time longer than creating the whole experiment in the prototype. In the end, they failed at creating the branching because PCL had to be used. Two more participants wrote in addition a basic approach in PCL, but the given time was far too short for both to finish the task. These two needed 30 % more time for programming the simple part than in the prototype. Finally, one user managed to accomplish the whole experiment with PCL. He was also the only one who developed the simple scenario part in shorter time than the whole experiment in the prototype (20 % less). However, for implementing the whole task in Presentation he needed 20 % more time.

Needed much more time for simple part

One succeeded: Branching clearly took more time

The reasons Presentation did poorly are manifold:

Several problems with Presentation: Shortcomings in structure, syntax, parameters, PCL, etc.

- Most users had great difficulties with the structure of Presentation. They did not know where which parameters had to be used. Partly, they even had problems with the general concept of scenario objects, e. g. that bitmaps are parts of pictures or how trials are constructed.
- The names and meanings of parameters were unknown in many cases and had to be looked up. Often the possible parameter values were not known, too.
- It was laborious to type every statement on their own instead of clicking it.
- The users had many syntax problems.
- Some users implemented the scenario unsystematically. This led to loss of time.
- Users spent much time with reading in the documentation.
- It was difficult to design the layout of the stimulus objects as it is not directly visible.
- Most users had great deficiencies in PCL.

The realization in Presentation was hard, although everyone used the existing script that was provided to them. It would have been simple just to copy some of the structures, but for some users it was difficult to understand the content at all.

Work with Visual Editor much more comfortable

The fulfilling of the task in the Presentation Visual Editor was much more comfortable for the users and they had less problems in the process. Just the problems that made Presentation difficult, did not occur with the prototype. For instance, no syntax errors are possible, the visual structure is more intuitive, stimulus layouts can be directly designed, and users had to consult hardly ever the documentation.

Users were able to handle the prototype well

During the introductory and the main task, the users handled the prototype well. They found all functions they needed in short time. None of them had problems with the creation of stimuli and they liked the direct manipulation and the drag and drop functionality. They were able to build the image trial quickly and enjoyed designing its layouts, e. g. they tried different colors and image sizes. The trial construction and the time line interactions also worked well. In the main screen, they made positive comments about the visibility of the structure and enjoyed the interaction with the tree. In particular, they appreciated that the trials can be freely arranged and that alignment guidelines are shown meanwhile. The users understood the creation of connections and their conditions quite fast and most of them also understood the general concept immediately. Only a few users had to think about how to create the branching at the end, but they found the solution on their own.

Usability issues and suggestions for improvement that I noted down during the test or that were stated in the questionnaire, are treated later in this section.

8.5.1 Questionnaire: General Impressions

Users preferred Visual Editor to Presentation

All participants enjoyed working with the prototype. Everyone fully agreed, that they like the Presentation Visual Editor and that they would prefer it to Presentation (median of 5 for both in the Likert scale).

Clearer, simpler, more user-friendly, and easier comprehensible

They enjoyed the general design of the Presentation Visual Editor (median 5). They said that they would change to a final system (median 5) and stated as reasons that it is easier comprehensible, clearer, simpler, and more user-friendly. Further advantages are that the parameter names are visible and that no syntax and programming is needed.

Saves time

They were convinced that the system would save lots of time and strongly agreed that it could make their work more efficient and faster (median 5). In particular, they strongly agreed that the editing time for creating experiments would be shorter than with Presentation (median 5).

More suitable for beginners

The users expressed that a switch to the Presentation Visual Editor would not be too laborious as they strongly disagreed that the time needed for familiarizing with it after a change from Presentation would be too long (median 1). All of the subjects also found that the system is more suitable for beginners, because the time for familiarization would be shorter than with Presentation (median 5).

Less errors and problems

The participants expected that they would make less mistakes when working with the Presentation Visual Editor (median 4) and that they would meet less problems when creating experiments (median 5).

8.5.2 Questionnaire: Time-Based Decision Trees

Quick comprehension of time-based decision trees

Half of the subjects were undecided whether they had problems to understand the visual tree structure immediately (median 3). This is most likely due to the fact that decision trees were unfamiliar to them, and thus, some users initially had slight problems. However, no one stated that he did not understand them and the initial problems could be overcome soon as they said that they were able to comprehend quickly how the tree structure works (median 4).

No interaction problems with the tree

No subject stated interaction problems with the trees in general; almost all disagreed with the corresponding statement (median 2). They also dissented that they had interaction problems with the connection arrows or with setting their conditions (median 2). However, four users either were undecided or admitted problems. They criticized that it was hard to perceive where trials can be connected and therefore they needed multiple tries. The majority of users did not have this problem.

Trees increase clarity of experiments

The participants' opinions about the tree structure were positive. They enjoyed constructing experiments visually and said that the visual tree structure helps with understanding the experiment's function and that it increases the clarity of experiments compared to Presentation scripts (all three statements have median 5).

8.5.3 Questionnaire: GUI

General preference for GUI

Except for one user all agreed with the statement that they would prefer a GUI for their work with Presentation (median 4) and hence disagreed with preferring text-based interaction (median 2). These questions were asked before they had seen the prototype. Afterwards, they said that Presentation generally benefits from graphical interaction (median 5). The deviant from before was neutral in this question. Most subjects saw no disadvantages when using a GUI (median 2). Four of them, however, were undecided and questioned whether a GUI can provide equal possibilities in comparison with the text editor and were concerned about losing control about some scenario details.

The participant who contradicted the others said that he prefers code-writing in general. But even he admitted that he was pleasantly surprised, because the Visual Editor has clear advantages when something should be tried quickly and because users do not have to struggle with syntax.

Use of both programs neutralizes their drawbacks

Since the Visual Editor is not intended to be an independent experiment interface, it was interesting to learn that the subjects think that the combination of the Visual Editor and Presentation for the creation of experiments would neutralize possible drawbacks of textual and graphical interaction, and instead it would combine their advantages (median 5). In this context, the subjects were contrary to the statement that the Visual Editor would increase the complexity of work as two programs have to be used (median 1).

8.5.4 Questionnaire: Trial Window

Users enjoyed stimulus creation and manipulation

The results concerning stimuli are also very clear. The participants liked the creation of visual stimuli in the prototype (median 5). Creating and editing the stimuli was intuitive and natural to them (median 4). They liked that they were able to drag and drop images directly into the program (median 4). And they also really liked the possibility of direct manipulation of visual stimuli (median 5). Finally, they almost unanimously agreed to the full extent to the statement that they liked that all changes to visual stimuli were directly visible (median 5).

Users enjoyed the time line

All participants really enjoyed the time line in the trials (median 5). The time line facilitated to get an overview of the course of events within a trial (median 5), and no one had mentionable interaction problems with it (median 4).

8.5.5 Questionnaire: Complex Tree

Users understood complex tree

At the end, everyone had fully understood the concept of time-based decision trees. All subjects were able to explain the complex example scenario correctly to me and stated in the questionnaire that they have understood it quickly and without any problems (median 5). In the meantime, they also felt confident with the prototype's operation. They said that they would dare to construct a similar experiment with the Visual Editor on their own (median 5).

Would dare to construct similar experiment with Visual Editor

Would not dare with Presentation

This was completely different with Presentation. Almost all disagreed that they feel confident to be able to construct a similar experiment with Presentation (median 2). Only the three participants that had some PCL experience felt confident (median 4). However, these three, like everybody else, strongly disagreed that constructing the experiment with Presentation would not take longer than with the Visual Editor (median 1). Finally, all subjects contradicted that it would not be more complex to comprehend a similar time structure in a Presentation-script (median 1).

Would take longer in Presentation

8.5.6 Suggestions for Improvement

Several problems with the system became apparent, too. The users made various suggestions for improvement.

- To some users it was not entirely clear from where to where connection arrows can be dragged and where on the trial blocks they have to be fixed. At the moment, the arrows snap to possible connection points that are near. To solve the problem, the anchor points should be visualized in addition.

Connection anchor points should be visualized

- One user had reservations to the tree concept, because it is non-standard to common Windows applications.
- Some of the users were bothered by the need to press the enter key sometimes to process their input. For instance, when entering image scale values, the users have to press enter afterwards to initiate the computation of the other image values. Only then, clicking OK leads to a changed image size. In the header dialog problems with the enter key occurred, too, because it was not obvious to some users. They suggested that the typing of values already initiates further processes.

Need for enter was bothering

- One user suggested to add tool tips to input fields that should display the possible parameter values.

Tool tips for input fields

- Two users mentioned that the initial arrangement of stimulus objects could be more sophisticated.
- The prototype lacks a coordinates display for precise stimulus positioning or some kind of rulers.

Coordinates are not displayed

- Direct enlargement and downsizing of stimuli still have some bugs.
- It should be possible to move multiple stimuli at once.
- Two users suggested to add a function for defining simple graphical, geometrical stimuli (boxes, cylinders, etc.). This would be a Presentation function that would clearly benefit from visual interaction.

Editor for geometrical shapes

Tool tips for time line

- The time line would benefit from tool tips that display the exact duration of an event.

Time line too sensitive

- The time line appeared to be too sensitive in some cases, which caused accidental value changes.

Events cannot be deleted

- A function for deleting stimulus events is missing.
- Several users emphasized that auditory stimuli are second to visual stimuli in their experiments and need to be implemented next.

Study reached its goal

All in all, the Presentation Visual Editor performed better than Presentation in direct comparison. The participants understood its concept quickly and clearly preferred the whole interface to Presentation. The prototype enabled the users to create the experiment in less time and with less problems. The users already needed much less time for realizing the simple part of the scenario. But the superiority and the benefits of the time-based decision trees became particularly apparent as soon as some complexity had to be overcome. While all participants successfully created the branching within short time in the prototype, only one user succeeded at all in Presentation—with more effort. Most users did not even come close to the solution.

Chapter 9

Conclusion

"The mind is never satisfied with the objects immediately before it, but is always breaking away from the present moment, and losing itself in schemes of future felicity... The natural flights of the human mind are not from pleasure to pleasure, but from hope to hope."

—Samuel Johnson

This work described the design process of the Presentation Visual Editor, an alternative interface for the software system Presentation. In contrast to the original interface, the whole interaction is visual. It is a development tool for the creation of psychological experiments without the need for programming. Time-based decision trees were incorporated into the interaction design to model the structure of experiments visually. The creation and the design of stimulus presentations were facilitated with visual interactions that enable users to directly manipulate stimulus layouts.

I started the design process with a Contextual Inquiry, which was conducted to get an understanding of the psychological domain and to identify problems in the operation of Presentation. Thereafter, the design evolved in three cycles of an iterative user-centered design process. The final prototype was evaluated in an additional user study that compared it with Presentation to verify the design.

Prototype not suitable for productive use

The end product of this work is still a prototype. Though it is an evaluated working system, it is not suitable for productive use, yet. While it already includes many important features, other crucial elements are only simulated or are completely omitted. The fundamental design has proven to be intuitive and effective so that the interface and the interactions should not change considerably anymore. However, many functions have to be added in the future to complete the system.

Several open issues

During the various evaluations of the prototypes, several issues arose and suggestions were made by users that are still open problems. I omitted these issues for reasons of time or because they were out of the scope of this work and would not have contributed much to the results. These issues will be covered in this chapter and I will discuss them for future work. Furthermore, I will discuss the significance of this work for general interaction design. This chapter concludes with a summary of the most important points of this work.

9.1 Future Work

9.1.1 Extension to Full Experiment System

Extend to more stimulus types

The prototypes of the Presentation Visual Editor were limited to the creation of experiments consisting of visual stimuli. To become a fully productive system, the interface must be extended to support all stimulus types Presentation supports. The main focus should be on auditory and video stimuli, since they are used in many experiments.

Similar interactions for other stimulus types

The interactions for creating and managing those stimuli should not differ too much from the existing ones. For example, drag and drop of stimulus files should be possible for all kinds of stimuli. The basic interaction with the time line should not differ for different kinds of events anyway. However, the time line will become more complex with more stimulus types, since more overlapping stimulus events are possible. For instance, auditory and visual stimuli can be presented independent of each other within a trial with possible overlapping. While preview functions are more complex for other stimulus types, the general way of interacting with the stimuli should be adopted from the current design.

More overlapping events

Besides more stimulus types, another issue of great importance is the realization of fMRI-capable scenarios directly in the Visual Editor. The users suggested several times that the interface should support fMRI features visually, too.

Realize fMRI-capable scenarios

Those scanner experiments have more complex requirements for time. The stimulus presentation must be coordinated with the scanner pulse tacts. Further, it is necessary to define so-called jitter times. This is due to the BOLD response curve the fMRI measures (see chapter 2.1.6). It is not possible to predict the exact time of this response, and it is important to scan different regions of the BOLD response curve in the experiments. Therefore, it is necessary to apply varying delays, the jitters, for the scans. This jitter control for the scanner is difficult and is mostly done manually in Presentation.

Complex time requirements for fMRI

To support fMRI, the interface should provide functions for computing all factors. It would be necessary to take the durations of stimulus blocks, the pulse tacts, and the time the fMRI needs for scanning the brain (called TR) to calculate jitter times.

The existing time line could support the visualization of fMRI pulse control. The scanner pulses could be depicted in the time line and there could be functions for aligning stimuli to pulses and for specifying jitter times. Users also suggested to define tables of jitter values with time intervals and minimum intervals. These jitter tables could then apply jitters randomized to the trials. This would eliminate much manual work.

Visualize fMRI pulses

All in all, incorporating fMRI support into the Presentation Visual Editor is a complex issue that must involve own user studies to ensure usability.

Complex incorporation of fMRI

9.1.2 Realization of All Displayed Functions

To further complete the system, it is necessary to implement all functions that are visible in the interface but that do not have real functionality. I will consider the most important ones here.

Implement conversion into SDL/PCL code

First, a fully functional system definitely must be capable of converting the scenarios to executable SDL/PCL code. Since Presentation's structure was adopted in the system, the conversion into scenario objects will not be that hard. The construction of the scenario header will be even simple. More difficult is the conversion of the tree structures as soon as there are branchings, because then PCL code has to be generated. However, it is important to consider that not always PCL code should be generated. When SDL is sufficient, solely SDL should be generated, because otherwise the users would have unnecessary difficulties with understanding the scripts.

Implement load/save

Furthermore, for obvious reasons it is important to implement the load and save functions. It should be considered to use some database management system for making the objects persistent.

Implement multiple scenario management

Finally, the data structure has to be changed in some point to allow managing multiple scenarios. Therewith, the experiment project function, which is displayed in the main screen, could be realized.

9.1.3 Repeated Template Structures

Template function is important

A function for repeating trial sequences was included in the paper prototype, but it was not realized in the interactive software prototype. During the Contextual Inquiry and in several comments made by users during the evaluations, it became clear that the users appreciate the template function and that they find it indispensable for the Visual Editor, too. Experiments often contain large sections of repeated structures in which many stimuli and/or tasks are shown. Therefore, this part of the paper prototype should definitely be realized in future designs.

Better visual cues for line dragging

Some aspects of the design need to be improved, though. While the concept of specifying stimulus orders and combinations by dragging visual lines between the varying elements and a template table was liked by the users, all had initial problems to discover the function. In a future system, there must be better visual cues that lines can be dragged. I would consider to add some mouse over effects to the varying elements and to the columns of the table that show starting points of the lines. In addition, the lines should not be visible all the time. Maybe the visibility of lines should be controlled by mouse over functions as well.

More sophisticated randomization

Further, the template table should be extended by more sophisticated randomization functions. The paper prototypes provided only for simple randomization of all table lines. Users suggested to add conditional randomization, e. g. dependent on the values of one column.

Adopt visualization

Finally, in the main screen, the button for creating the repeated structures must be provided with functionality. The visualization of repeated structures in the decision tree should be simply adopted from the second paper prototype. For this feature it is necessary to introduce a function for selecting multiple trials at once. This should be done with common methods. Selecting multiple trials has also benefits for the normal tree interaction as this would allow to move them all at once. Thus, arranging the tree would become more comfortable.

Selection of multiple trials

9.1.4 Stimulus Libraries

Reintroduce stimulus libraries

The evaluation of the second paper prototype (see chapter 6.4) made clear that the users were for reintroducing the stimulus libraries. The need for a trial library in the main screen is not given anymore as the system includes a trial copy function. Stimulus libraries, however, could make experiment construction more efficient as they allow to reuse existing objects. In large scenarios with recurring stimuli, such as text objects for task specifications, this would eliminate the need to define the stimuli each time again.

Add import function

The libraries should be shown on request instead of occupying screen space all the time. A good solution might be to realize them in the form of a panel that is independent of a particular window. In addition, it is useful to add an import function for libraries from other experiments to fully exploit the possibilities of such a library.

9.1.5 Help and Error Prevention

Improve help system

In order to save users from frustration because of errors and wrong inputs and in order to save them the effort of reading in the Presentation documentation because they do not understand the meaning of functions or parameters, the help system of the Presentation Visual Editor should be improved. Besides a manual, that could be written, it is important to extend the use of tool tips, since they provide immediate help.

More tool tips

In the final prototype exist only few tool tips. Most of them explain the meaning of a button. However, users insisted already in the evaluations of the paper prototypes on more tool tips. There should be a tool tip for every parameter in the system that briefly explains the meaning of the parameter and maybe its possible values. This might not be sufficient for novices, but tool tips are ideally suitable as reminders and for quick overview.

Tool tips for input fields and to display information

In the final evaluation it was suggested to add tool tips to the input fields of parameters, too. This would allow users to learn what kind of values are expected for a particular parameter. Tool tips could also serve as information medium. In the time line they could be used to show the exact values of events, and in the layout area they could display properties like the name or the measurements of an image stimulus.

Incomplete error prevention

In addition, the error prevention in the system is incomplete, too. Parameters and input fields that would not make sense to be used are greyed out, but nothing is done so far to prevent users from entering invalid values. In future versions of the program the users should be advised of their mistakes.

9.1.6 Test Mode

Users want test mode

The users stated several times that they would like to have a test mode which provides a time line for running back and forth through experiments at different speeds and that would allow to halt experiments. Presentation itself allows only to run the experiments in real-time from the beginning. The debugging of experiments thus is complicated and time-consuming.

Realization seems impossible

However, I do not believe that it is possible to add such a test mode to the Presentation Visual Editor. In order to realize the function, it would be inevitable to access Presentation's compiler. This is not possible, though. A simulated test run in an own system would not help, since the execution in Presentation is crucial.

Disabling of trials to facilitate testing

During the evaluations another idea was expressed how to facilitate testing. The idea was to exclude certain trials from a scenario in order to be able to execute only parts of it. This test mode should be considered for implementation in a future system. Trials could be marked as disabled with the effect that they are ignored in the conversion of the scenario to Presentation code.

9.1.7 Significance for Interaction Design

How significant are the results for general interaction design?

This work has successfully shown that time-based decision trees are beneficial in the interaction design of psychological experiments in comparison with Presentation. The greater goal, this work pursues, is to show that time-based decision trees are generally beneficial in interaction design, since they efficiently allow to model time in interactions. The specialization on the psychological domain was chosen to test the concept in practice. The question now arises to what extent the positive results of the specialized case can be generalized to interaction design on the whole, i. e. how significant are the results for interaction design.

Results cannot be transferred

Basically, the results cannot simply be transferred to general interaction design for various reasons. Firstly, the de-

Very specialized case of interaction design

sign of psychological experiments is a *very* specialized case of interaction design. It can be questioned whether this really reflects a typical case of interaction design—in particular, because the interaction designers, the psychologists, are no programmers or software engineers and do not want to become some.

Success of Visual Editor was not caused solely by time-based decision trees

There is also the fact that the Presentation Visual Editor outperformed Presentation not solely due to time-based decision trees and their capability of concrete visual modeling of time structures. The Visual Editor took also advantage of its pure visual interaction. It performed better, because no programming is needed in experiment construction and because stimulus creation and editing with direct manipulation is much simpler and more comfortable than in Presentation. Furthermore, the user interface of the reference system Presentation has many usability problems and is disliked by its users.

Thus, it can be debated whether the results would have been similar clear, if the reference system were good (without modeling of time, though) and the users were programmers who do not have fundamental problems with the structure of the program. All those reasons make direct generalization of the findings impossible.

Results motivate further research

However, the impressions of the work with time-based decision trees are very positive and give motivation to further research the concept. Nothing in the results suggests negative aspects of the tree concept, instead all results were so clearly positive that the concept should definitely be pursued in further projects.

Concept proved to be efficient, easily readable, and easily comprehensible

First of all, the results showed that time-based decision trees are capable of modeling time structures of interactions efficiently. At the same time, they are easily readable by humans. Throughout the evaluations, no participant had problems to comprehend the structure of experiments or the general structure of the graphs. The concept of time-based decision trees in the experiment/interaction design was always understood quickly.

All in all, this work cannot draw a conclusion for the benefits of time-based decision trees in general interaction design. But this project delivered promising results that should motivate future projects.

9.2 Summary

The goal of this work was to overcome the limitations in current software development in the design of the temporal layout of interactive systems. I proposed a solution for modeling time in interactions that can be integrated in the design process of interactive systems: time-based decision trees. I wanted to show that this concept is beneficial in interaction design.

The approach was proved by designing a visual development system for the creation of psychological experiments. In three iterations of user-centered design I developed an innovative and working system based on time-based decision trees. The final prototype allows to construct experiment structures and all experiment components with visual interactions. Thereby it facilitates the design of time structures in experiments by providing efficient interactions with time-based decision trees.

The final evaluation of the system delivered qualitative and quantitative results that show that the design of the Presentation Visual Editor is superior to a conventional system. I was able to verify the benefits of time-based decision trees in the design of psychological experiments. The users understood the concept and applied it effectively.

It was not possible to prove the usefulness of the concept in general interaction design, since the examined application domain is very specialized. This is an open issue left for future research.

Appendix A

Additional Paper Prototype Screens

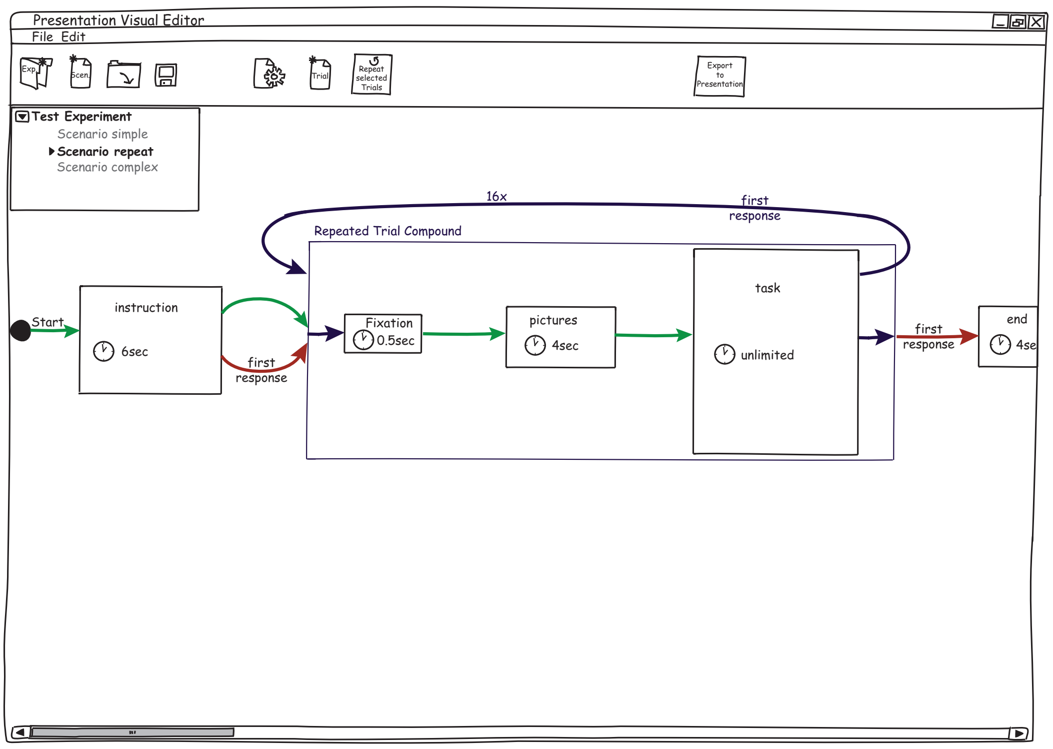


Figure A.1: Main screen of the second prototype showing an example experiment, which was used in the evaluations. It contains a repeated trial compound visualized as one block. The middle three trials are repeated 16 times.

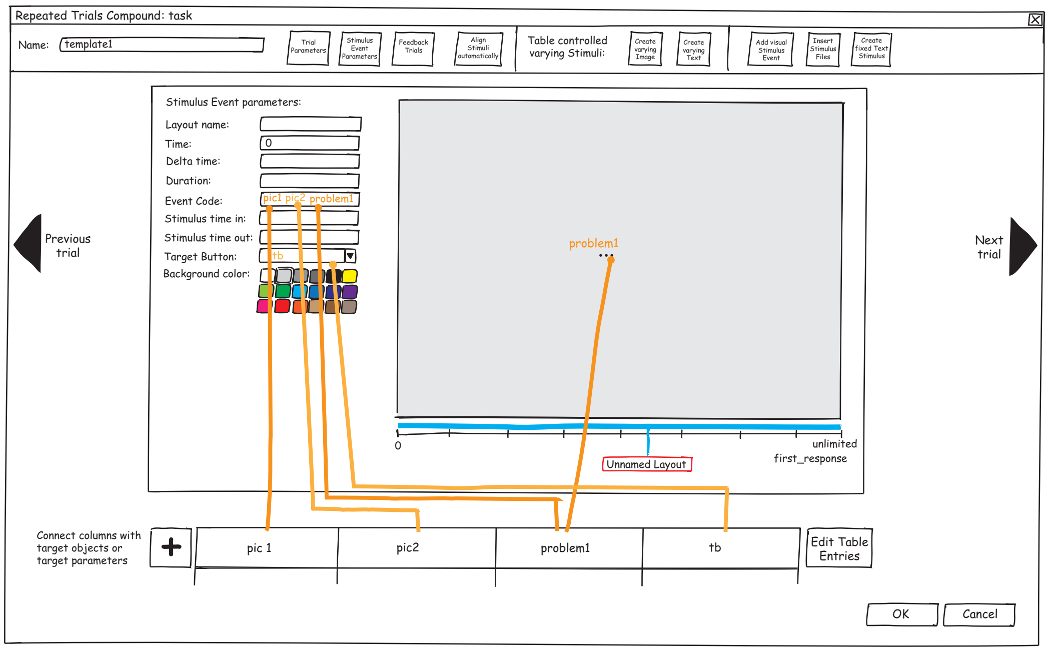
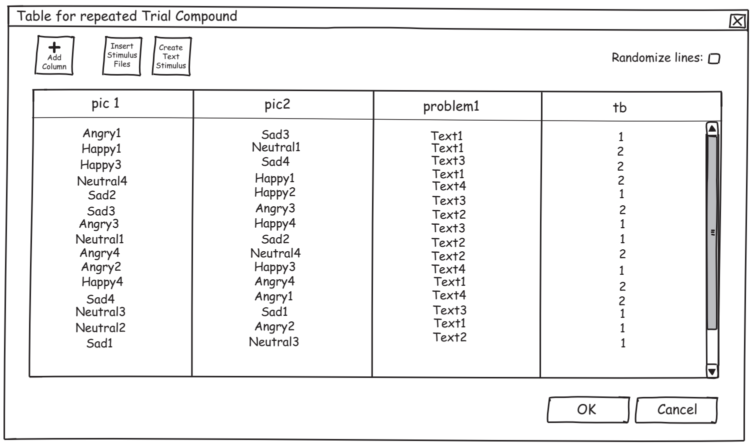
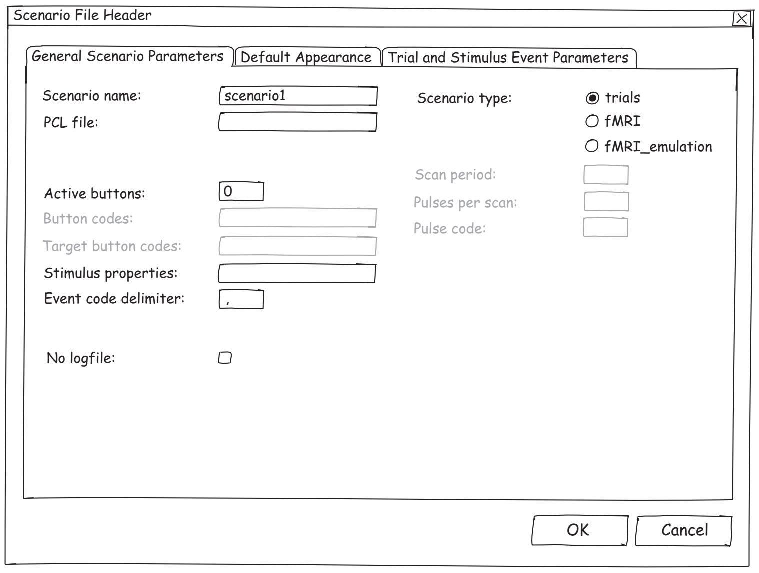


Figure A.2: Repeated trial compound window from the second prototype for managing trials contained in template structures. In the shown stimulus event, a different text stimulus is presented in every iteration. The event code parameter is connected with three columns. This has the effect that the names of all stimuli, that were presented in an iteration, are written to the log file. Finally, the target button parameter is connected with a column to define a different correct response button for the stimulus event in every iteration.



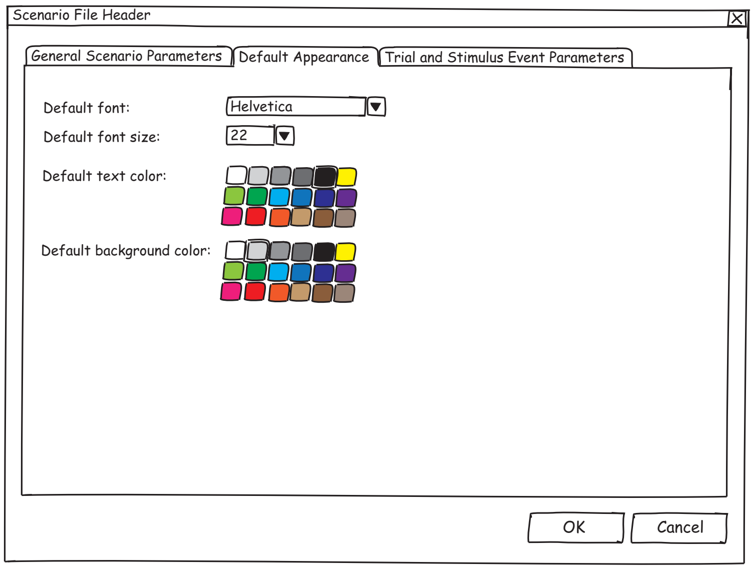
pic 1	pic2	problem1	tb
Angry1	Sad3	Text1	1
Happy1	Neutral1	Text1	2
Happy3	Sad4	Text3	2
Neutral4	Happy1	Text1	2
Sad2	Happy2	Text4	1
Sad3	Angry3	Text3	2
Angry3	Happy4	Text2	1
Neutral1	Sad2	Text3	1
Angry4	Neutral4	Text2	2
Angry2	Happy3	Text2	1
Happy4	Angry4	Text4	2
Sad4	Angry1	Text1	2
Neutral3	Sad1	Text4	1
Neutral2	Angry2	Text3	1
Sad1	Neutral3	Text1	1
		Text2	

Figure A.3: This window from the second prototype shows an example template table for a repeated trial compound.


Scenario File Header
General Scenario Parameters
Default Appearance
Trial and Stimulus Event Parameters
OK
Cancel

Scenario name: scenario1
PCL file:
Active buttons: 0
Button codes:
Target button codes:
Stimulus properties:
Event code delimiter: ,
No logfile: ☐
Scenario type: [✓ trials] fMRI fMRI_emulation
Scan period:
Pulses per scan:
Pulse code:

Figure A.4: Header dialog for specifying general scenario parameters.


Scenario File Header
General Scenario Parameters
Default Appearance
Trial and Stimulus Event Parameters
OK
Cancel

Default font: Helvetica
Default font size: 22
Default text color:
Default background color:

Figure A.5: Header dialog for specifying parameters concerning the default appearance of stimuli.

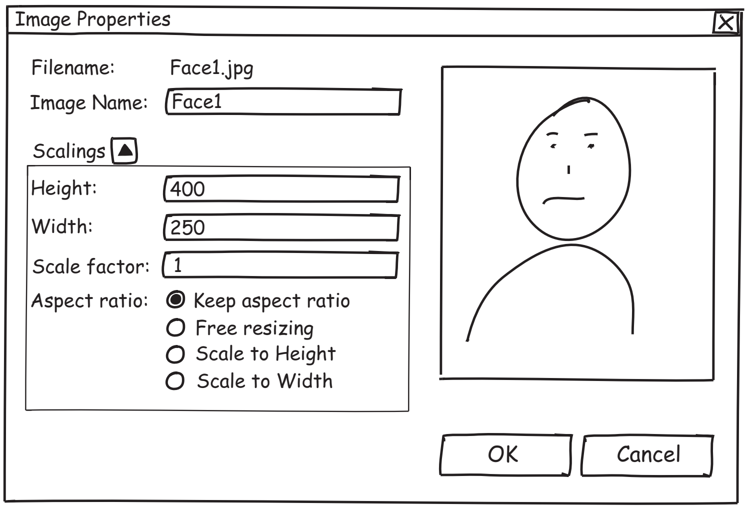


Image Properties

Filename: Face1.jpg

Image Name: Face1

Scalings

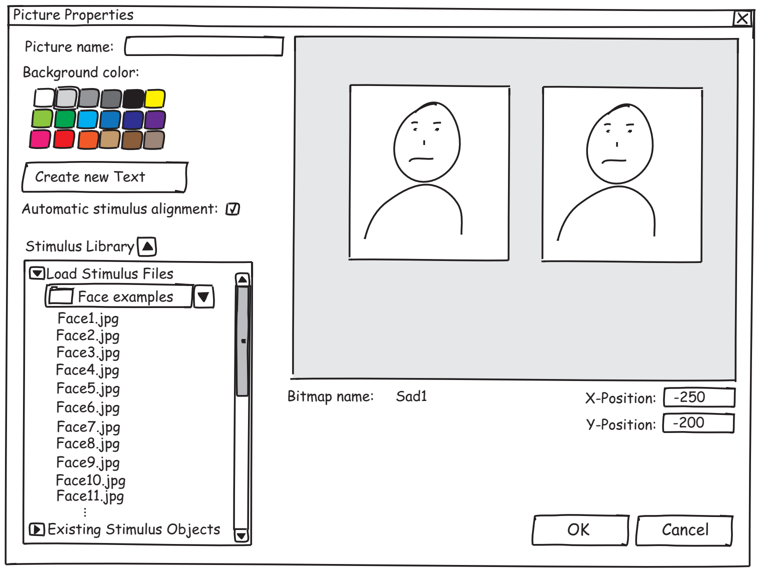
Height: 400

Width: 250

Scale factor: 1

Aspect ratio: [✓ Keep aspect ratio] Free resizing Scale to Height Scale to Width

Figure A.6: Image properties dialog from the second prototype showing an image and its measurements.


Picture Properties

Picture name:

Background color:

Automatic stimulus alignment: ✓

Stimulus Library

Load Stimulus Files

Face examples

Face1.jpg
Face2.jpg
Face3.jpg
Face4.jpg
Face5.jpg
Face6.jpg
Face7.jpg
Face8.jpg
Face9.jpg
Face10.jpg
Face11.jpg
⋮

Existing Stimulus Objects

Bitmap name: Sad1

X-Position: -250

Y-Position: -200

Figure A.7: Picture properties dialog from the first prototype. It displays a picture object with two bitmap stimuli. Since it was quite redundant, it was withdrawn and its functions were integrated into the trial window.

Appendix B

Software Prototype: Sample Run

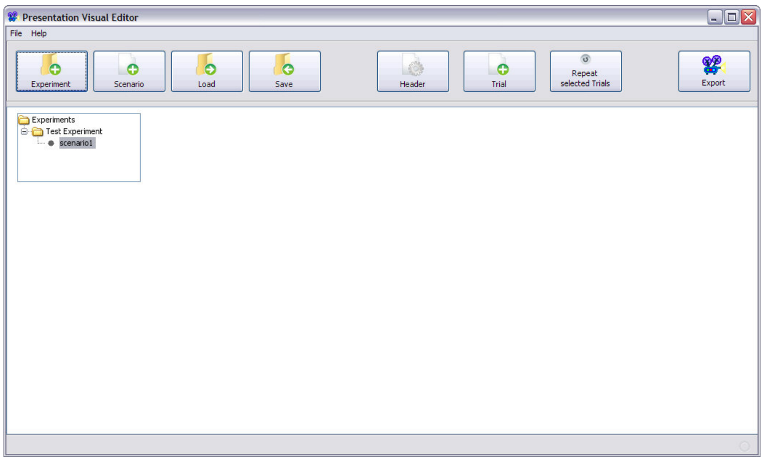
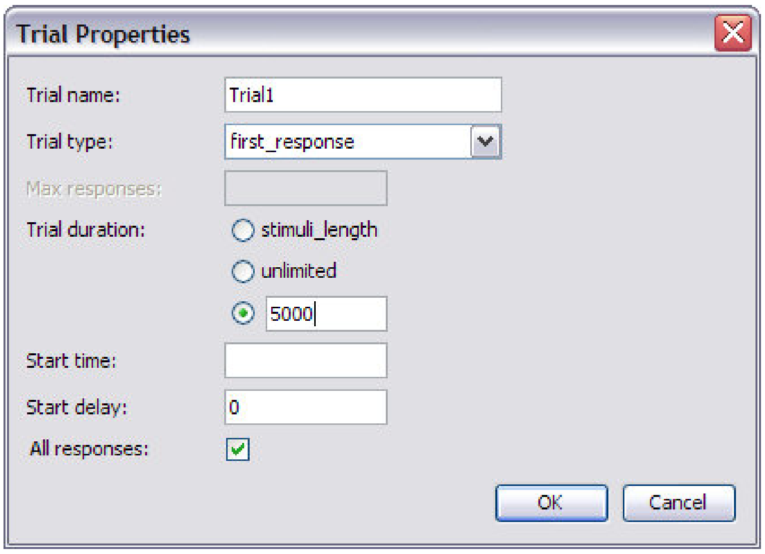


Figure B.1: After starting the application, the user sees the empty main screen. The current scenario is called *scenario1* and is contained in *Test Experiment*. These are created by default.



Trial name:	Trial1
Trial type:	first_response
Max responses:	
Trial duration:	stimuli_length [ ]
	unlimited [ ]
	[✓ 5000]
Start time:	
Start delay:	0
All responses:	✓

Figure B.2: After pressing the button for creating a new trial, this dialog appears. All trial parameters can be specified in here. The trial name is set to *Trial1* (default name). The trial can be ended by subject response with any button by selecting *first_response*. The (maximum) duration is set to 5 seconds. The other parameters' defaults are kept.

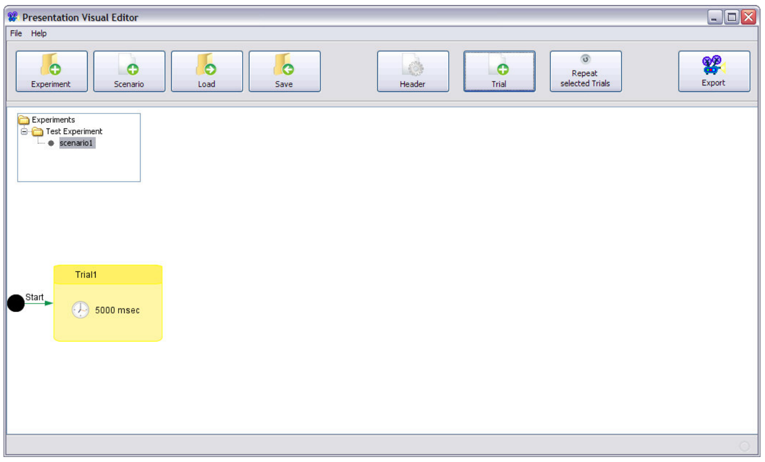


Figure B.3: The new trial was created by pressing *OK* and appears in the main screen. The trial block shows its name and duration. It is the starting trial in this scenario.

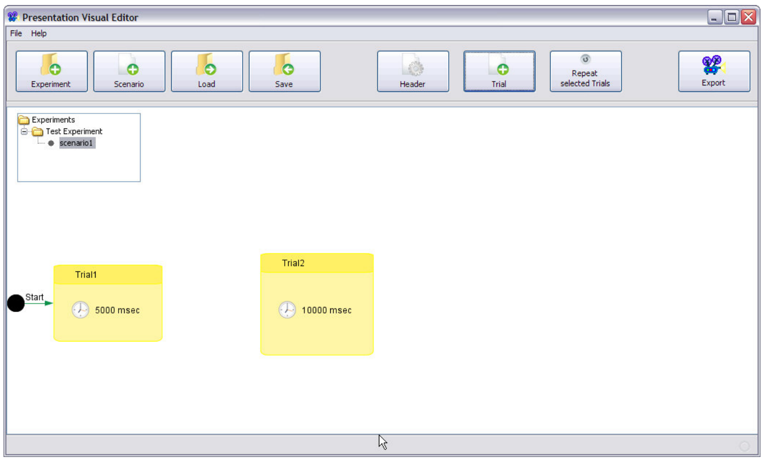


Figure B.4: In the same way, a second trial, *Trial2*, is created that lasts for 10 seconds. Its longer duration is also indicated by its bigger size.

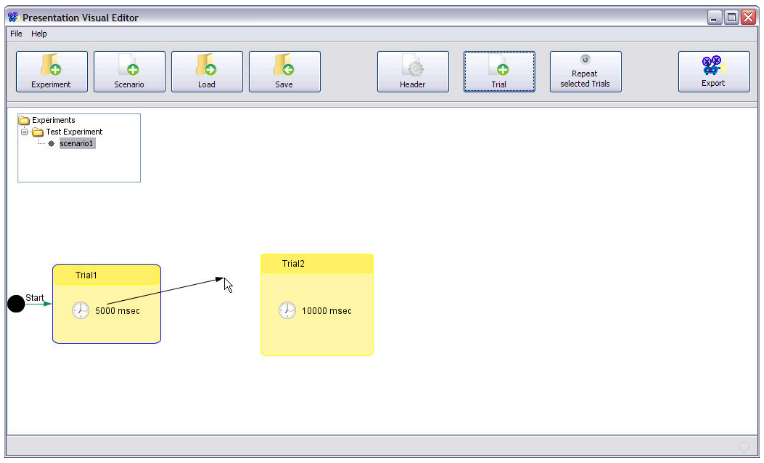
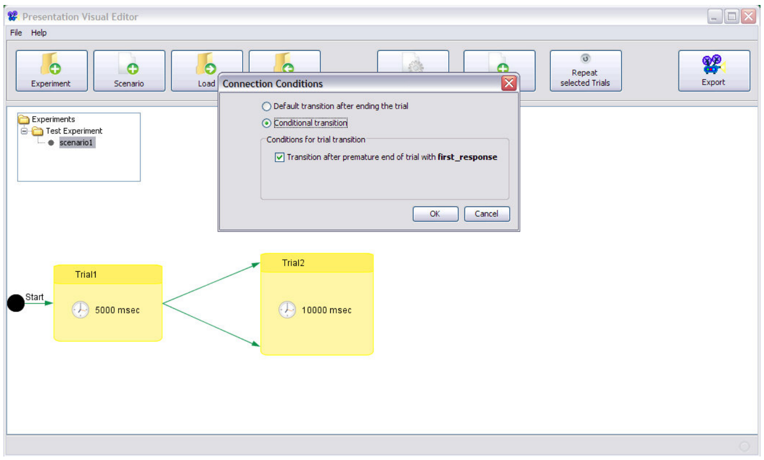


Figure B.5: A connection is dragged from *Trial1* to *Trial2*. To initiate the connection process, the user has to click anywhere in a trial while pressing the control key.



Connection Conditions

( ) Default transition after ending the trial
[✓ Conditional transition]

Conditions for trial transition

[✓] Transition after premature end of trial with **first_response**

OK Cancel

Figure B.6: Two connections were successfully created. One is kept as standard transition that is taken if the trial ends regularly after 5 seconds. After clicking the other one, a dialog appears. This connection is set to be taken after the subject has prematurely ended the trial.

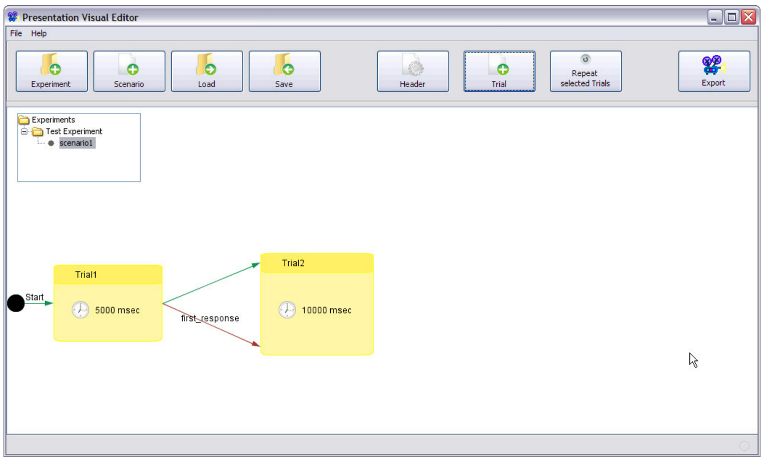


Figure B.7: The two connections are now fully defined. The conditional transition is displayed in red and is labeled with its condition.

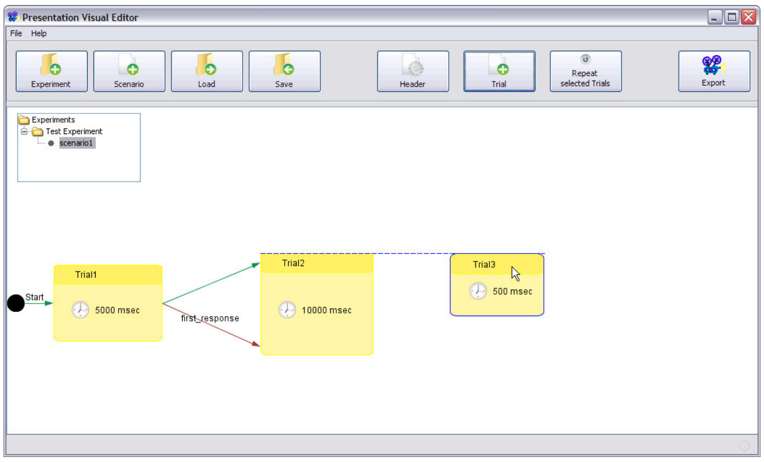


Figure B.8: A third trial was inserted. When its position is changed, alignment guidelines appear that allow a nice arrangement.

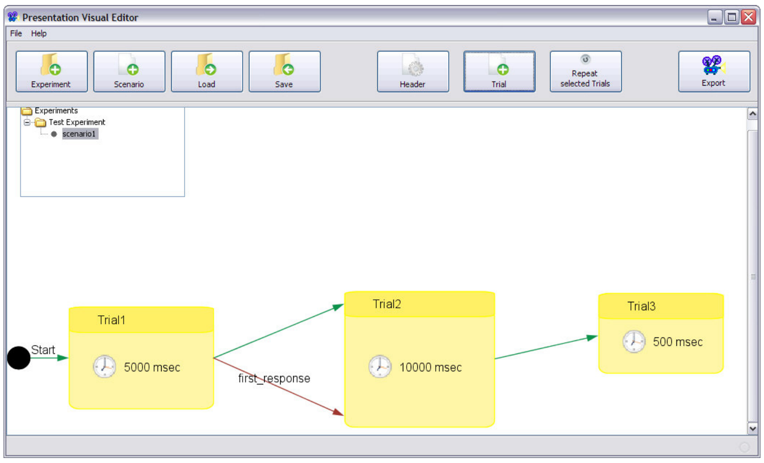
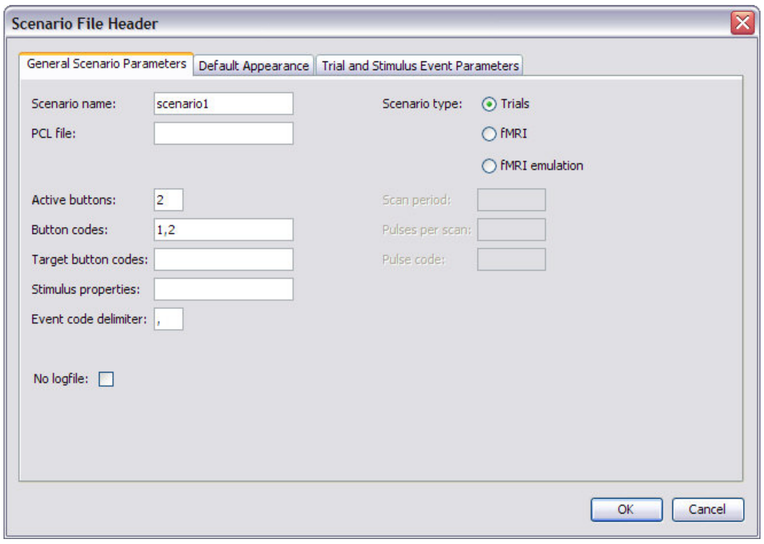


Figure B.9: The scroll wheel of the mouse (in combination with the control key) was used to zoom in the tree structure.


Scenario File Header
General Scenario Parameters
Default Appearance
Trial and Stimulus Event Parameters
OK
Cancel

Scenario name:	scenario1	Scenario type:	[✓ Trials] fMRI fMRI emulation
PCL file:			
Active buttons:	2	Scan period:	
Button codes:	1,2	Pulses per scan:	
Target button codes:		Pulse code:	
Stimulus properties:			
Event code delimiter:	,		
No logfile:	☐		

Figure B.10: After pressing the header button, a dialog for specifying the scenario header appears. It is used to define two response keys for the subject with the *Active buttons* parameter. The buttons are assigned the codes *1* and *2*.

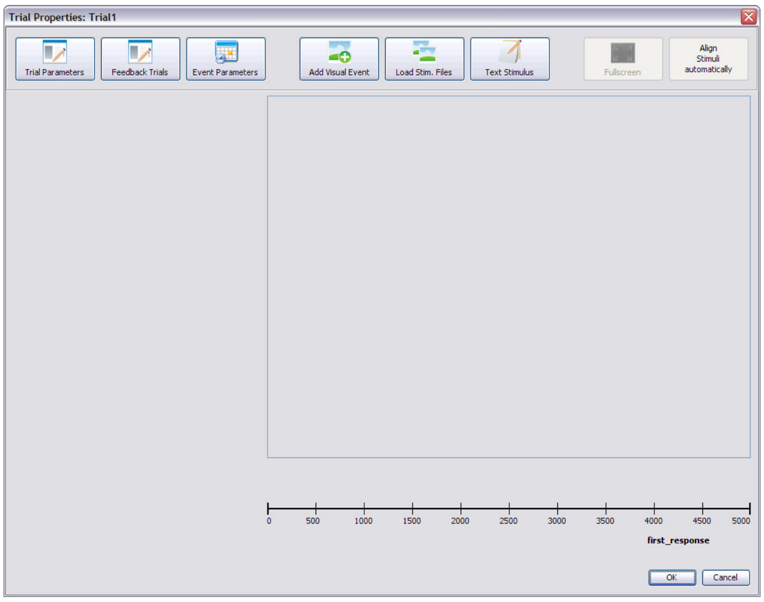
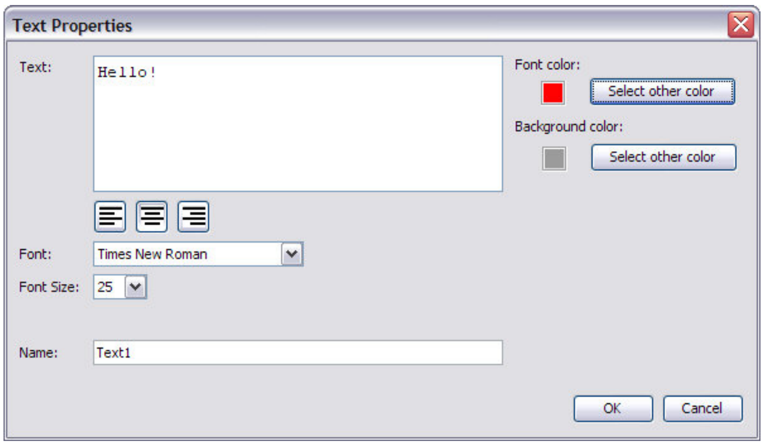


Figure B.11: After double-clicking *Trial1* in the main screen, this *Trial Properties* window opens. The trial contains no stimulus event, yet. Thus, the layout area is empty and nothing is displayed in the time line.


Text Properties

Text: Hello!

Font color: Select other color

Background color: Select other color

Font: Times New Roman

Font Size: 25

Name: Text1

OK Cancel

Figure B.12: After pressing the *Text Stimulus* button, this dialog opens. A text is entered and its font type, font size, and font color are specified.

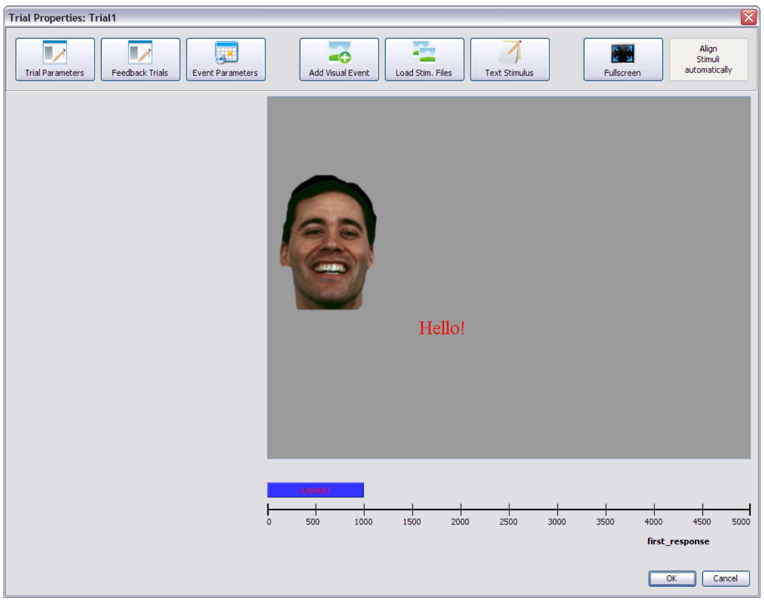


Figure B.13: A stimulus event (with default duration from header) and an associated layout were created automatically. The text stimulus is displayed in the layout. An image stimulus was created by dragging an image file from the Windows Explorer into the layout.

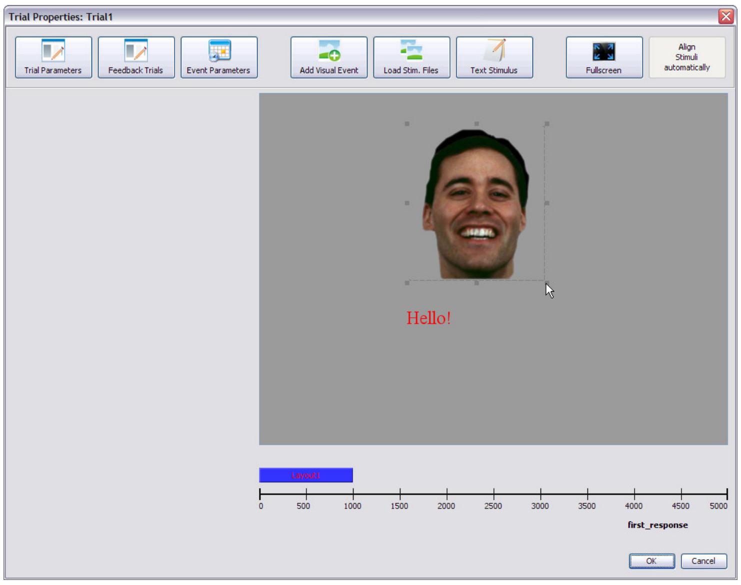
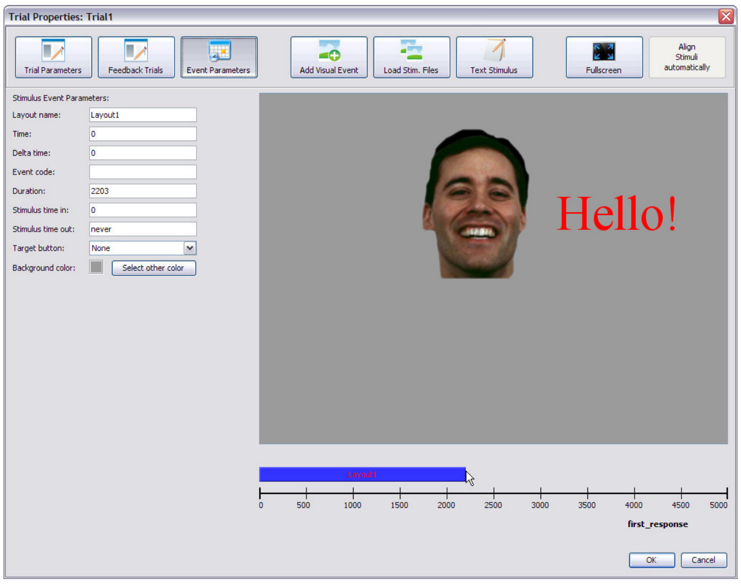


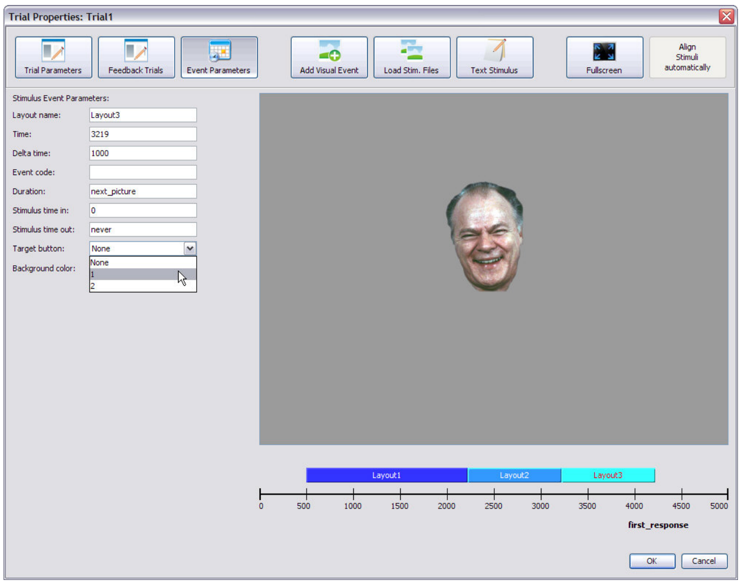
Figure B.14: The image was moved to the center. It is currently resized by dragging its lower-right corner. The borders are displayed when the cursor is over the object.



Stimulus Event Parameters:

Layout name:	Layout1
Time:	0
Delta time:	0
Event code:	
Duration:	2203
Stimulus time in:	0
Stimulus time out:	never
Target button:	None
Background color:	▨ Select other color

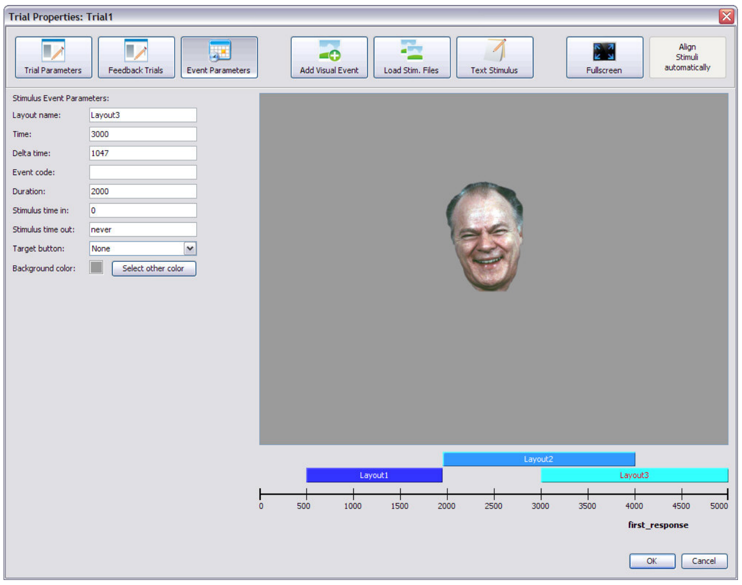
Figure B.15: The text was also moved and resized. In addition, the stimulus event parameters are now shown on the left. The event duration is currently changed by dragging the time line segment's endpoint to the right. The corresponding parameter value is updated appropriately.



Stimulus Event Parameters:

Layout name:	Layout3
Time:	3219
Delta time:	1000
Event code:	
Duration:	next_picture
Stimulus time in:	0
Stimulus time out:	never
Target button:	None
Background color:	None / 1 / 2

Figure B.16: By pressing the *Add Visual Event* button, two more events were added to the trial. The layout of the last event is shown. For this event a target button is specified. This is the correct button a subject has to press for the event. It can be chosen from the two previously defined buttons.



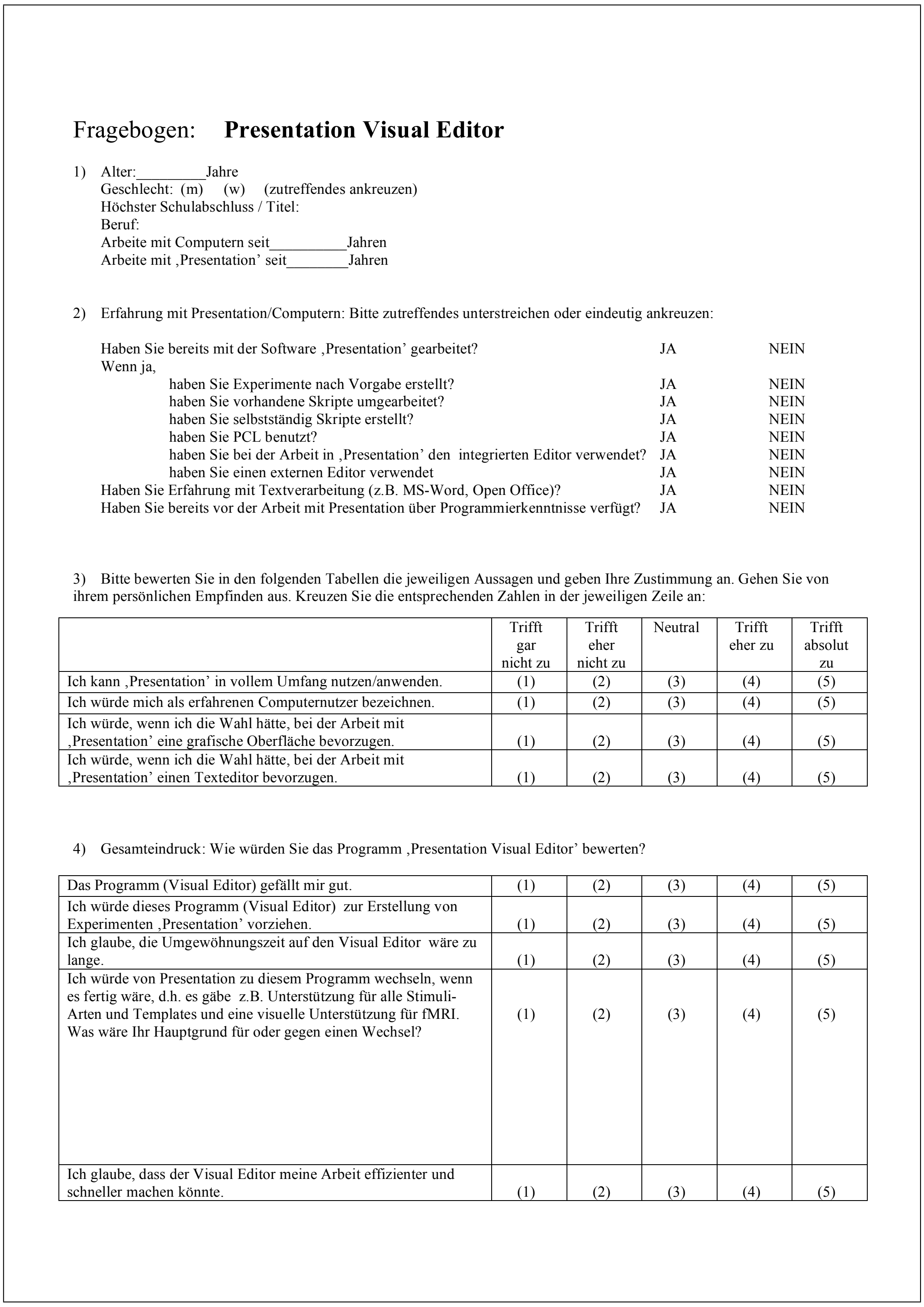
Stimulus Event Parameters:

Layout name:	Layout3
Time:	3000
Delta time:	1047
Event code:	
Duration:	2000
Stimulus time in:	0
Stimulus time out:	never
Target button:	None
Background color:	▨ Select other color

Figure B.17: The duration of the second event was made response-dependent. Therefore, it can overlap with the third event.

Appendix C

Final Evaluation: Questionnaire



Fragebogen: **Presentation Visual Editor**

1) Alter:_________Jahre
Geschlecht: (m) (w) (zutreffendes ankreuzen)
Höchster Schulabschluss / Titel:
Beruf:
Arbeite mit Computern seit__________Jahren
Arbeite mit ‚Presentation' seit________Jahren

2) Erfahrung mit Presentation/Computern: Bitte zutreffendes unterstreichen oder eindeutig ankreuzen:

Haben Sie bereits mit der Software ‚Presentation' gearbeitet?	JA	NEIN
Wenn ja,		
haben Sie Experimente nach Vorgabe erstellt?	JA	NEIN
haben Sie vorhandene Skripte umgearbeitet?	JA	NEIN
haben Sie selbstständig Skripte erstellt?	JA	NEIN
haben Sie PCL benutzt?	JA	NEIN
haben Sie bei der Arbeit in ‚Presentation' den integrierten Editor verwendet?	JA	NEIN
haben Sie einen externen Editor verwendet	JA	NEIN
Haben Sie Erfahrung mit Textverarbeitung (z.B. MS-Word, Open Office)?	JA	NEIN
Haben Sie bereits vor der Arbeit mit Presentation über Programmierkenntnisse verfügt?	JA	NEIN

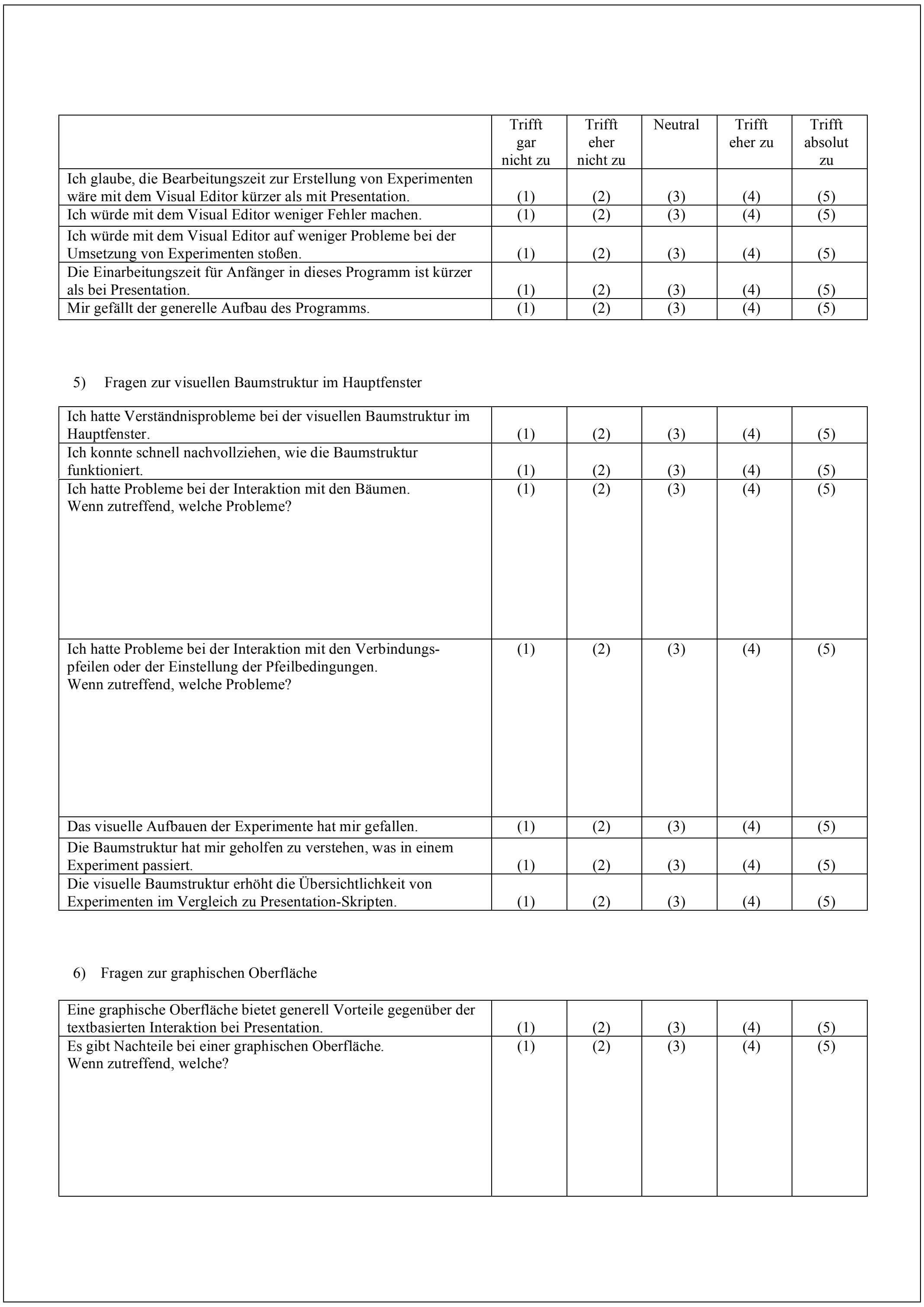
3) Bitte bewerten Sie in den folgenden Tabellen die jeweiligen Aussagen und geben Ihre Zustimmung an. Gehen Sie von ihrem persönlichen Empfinden aus. Kreuzen Sie die entsprechenden Zahlen in der jeweiligen Zeile an:

	Trifft gar nicht zu	Trifft eher nicht zu	Neutral	Trifft eher zu	Trifft absolut zu
Ich kann ‚Presentation' in vollem Umfang nutzen/anwenden.	(1)	(2)	(3)	(4)	(5)
Ich würde mich als erfahrenen Computernutzer bezeichnen.	(1)	(2)	(3)	(4)	(5)
Ich würde, wenn ich die Wahl hätte, bei der Arbeit mit ‚Presentation' eine grafische Oberfläche bevorzugen.	(1)	(2)	(3)	(4)	(5)
Ich würde, wenn ich die Wahl hätte, bei der Arbeit mit ‚Presentation' einen Texteditor bevorzugen.	(1)	(2)	(3)	(4)	(5)

4) Gesamteindruck: Wie würden Sie das Programm ‚Presentation Visual Editor' bewerten?

Das Programm (Visual Editor) gefällt mir gut.	(1)	(2)	(3)	(4)	(5)
Ich würde dieses Programm (Visual Editor) zur Erstellung von Experimenten ‚Presentation' vorziehen.	(1)	(2)	(3)	(4)	(5)
Ich glaube, die Umgewöhnungszeit auf den Visual Editor wäre zu lange.	(1)	(2)	(3)	(4)	(5)
Ich würde von Presentation zu diesem Programm wechseln, wenn es fertig wäre, d.h. es gäbe z.B. Unterstützung für alle Stimuli-Arten und Templates und eine visuelle Unterstützung für fMRI. Was wäre Ihr Hauptgrund für oder gegen einen Wechsel?	(1)	(2)	(3)	(4)	(5)
Ich glaube, dass der Visual Editor meine Arbeit effizienter und schneller machen könnte.	(1)	(2)	(3)	(4)	(5)

Figure C.1: The questionnaire (page 1) which the subjects were asked to fill out in the final evaluation of the interactive software prototype. The testers filled out 1)–3) in advance, the remainder at the end after doing the experiment.



	Trifft gar nicht zu	Trifft eher nicht zu	Neutral	Trifft eher zu	Trifft absolut zu
Ich glaube, die Bearbeitungszeit zur Erstellung von Experimenten wäre mit dem Visual Editor kürzer als mit Presentation.	(1)	(2)	(3)	(4)	(5)
Ich würde mit dem Visual Editor weniger Fehler machen.	(1)	(2)	(3)	(4)	(5)
Ich würde mit dem Visual Editor auf weniger Probleme bei der Umsetzung von Experimenten stoßen.	(1)	(2)	(3)	(4)	(5)
Die Einarbeitungszeit für Anfänger in dieses Programm ist kürzer als bei Presentation.	(1)	(2)	(3)	(4)	(5)
Mir gefällt der generelle Aufbau des Programms.	(1)	(2)	(3)	(4)	(5)

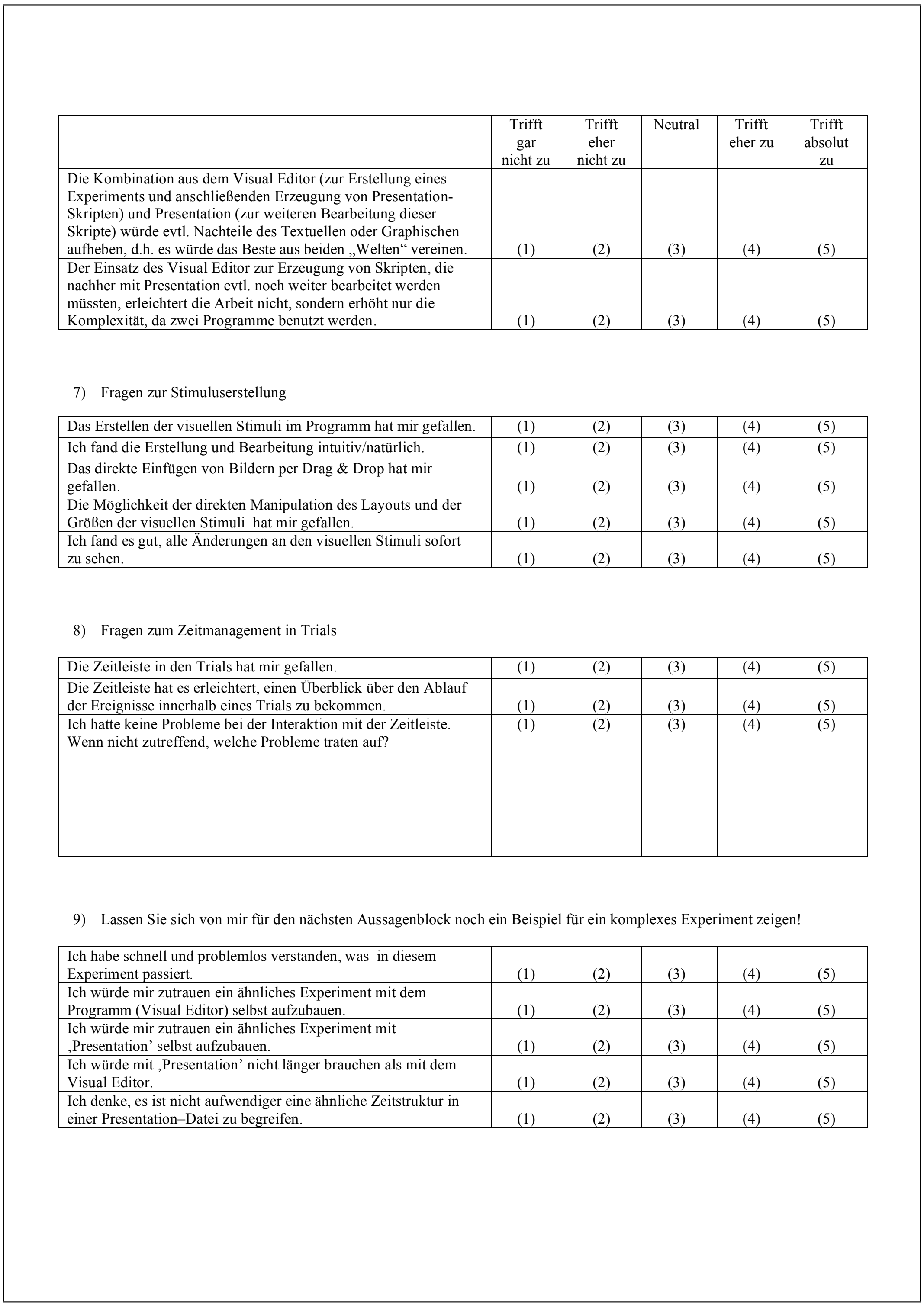
5) Fragen zur visuellen Baumstruktur im Hauptfenster

Ich hatte Verständnisprobleme bei der visuellen Baumstruktur im Hauptfenster.	(1)	(2)	(3)	(4)	(5)
Ich konnte schnell nachvollziehen, wie die Baumstruktur funktioniert.	(1)	(2)	(3)	(4)	(5)
Ich hatte Probleme bei der Interaktion mit den Bäumen. Wenn zutreffend, welche Probleme?	(1)	(2)	(3)	(4)	(5)
Ich hatte Probleme bei der Interaktion mit den Verbindungs-pfeilen oder der Einstellung der Pfeilbedingungen. Wenn zutreffend, welche Probleme?	(1)	(2)	(3)	(4)	(5)
Das visuelle Aufbauen der Experimente hat mir gefallen.	(1)	(2)	(3)	(4)	(5)
Die Baumstruktur hat mir geholfen zu verstehen, was in einem Experiment passiert.	(1)	(2)	(3)	(4)	(5)
Die visuelle Baumstruktur erhöht die Übersichtlichkeit von Experimenten im Vergleich zu Presentation-Skripten.	(1)	(2)	(3)	(4)	(5)

6) Fragen zur graphischen Oberfläche

Eine graphische Oberfläche bietet generell Vorteile gegenüber der textbasierten Interaktion bei Presentation.	(1)	(2)	(3)	(4)	(5)
Es gibt Nachteile bei einer graphischen Oberfläche. Wenn zutreffend, welche?	(1)	(2)	(3)	(4)	(5)

Figure C.2: The questionnaire (page 2) which the subjects were asked to fill out in the final evaluation of the interactive software prototype after doing the experiment.



	Trifft gar nicht zu	Trifft eher nicht zu	Neutral	Trifft eher zu	Trifft absolut zu
Die Kombination aus dem Visual Editor (zur Erstellung eines Experiments und anschließenden Erzeugung von Presentation-Skripten) und Presentation (zur weiteren Bearbeitung dieser Skripte) würde evtl. Nachteile des Textuellen oder Graphischen aufheben, d.h. es würde das Beste aus beiden „Welten“ vereinen.	(1)	(2)	(3)	(4)	(5)
Der Einsatz des Visual Editor zur Erzeugung von Skripten, die nachher mit Presentation evtl. noch weiter bearbeitet werden müssten, erleichtert die Arbeit nicht, sondern erhöht nur die Komplexität, da zwei Programme benutzt werden.	(1)	(2)	(3)	(4)	(5)

7) Fragen zur Stimuluserstellung

Das Erstellen der visuellen Stimuli im Programm hat mir gefallen.	(1)	(2)	(3)	(4)	(5)
Ich fand die Erstellung und Bearbeitung intuitiv/natürlich.	(1)	(2)	(3)	(4)	(5)
Das direkte Einfügen von Bildern per Drag & Drop hat mir gefallen.	(1)	(2)	(3)	(4)	(5)
Die Möglichkeit der direkten Manipulation des Layouts und der Größen der visuellen Stimuli hat mir gefallen.	(1)	(2)	(3)	(4)	(5)
Ich fand es gut, alle Änderungen an den visuellen Stimuli sofort zu sehen.	(1)	(2)	(3)	(4)	(5)

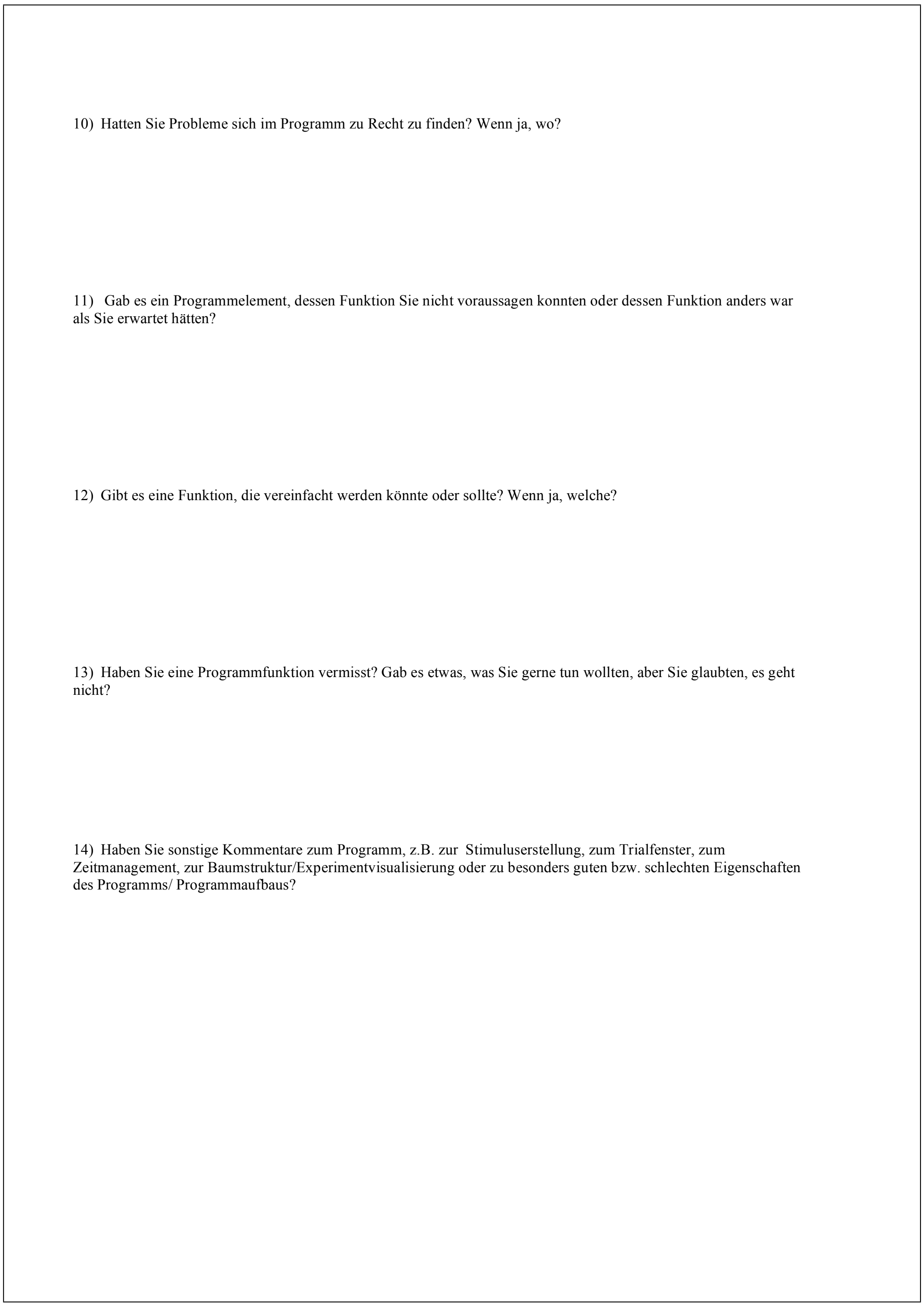
8) Fragen zum Zeitmanagement in Trials

Die Zeitleiste in den Trials hat mir gefallen.	(1)	(2)	(3)	(4)	(5)
Die Zeitleiste hat es erleichtert, einen Überblick über den Ablauf der Ereignisse innerhalb eines Trials zu bekommen.	(1)	(2)	(3)	(4)	(5)
Ich hatte keine Probleme bei der Interaktion mit der Zeitleiste. Wenn nicht zutreffend, welche Probleme traten auf?	(1)	(2)	(3)	(4)	(5)

9) Lassen Sie sich von mir für den nächsten Aussagenblock noch ein Beispiel für ein komplexes Experiment zeigen!

Ich habe schnell und problemlos verstanden, was in diesem Experiment passiert.	(1)	(2)	(3)	(4)	(5)
Ich würde mir zutrauen ein ähnliches Experiment mit dem Programm (Visual Editor) selbst aufzubauen.	(1)	(2)	(3)	(4)	(5)
Ich würde mir zutrauen ein ähnliches Experiment mit ‚Presentation’ selbst aufzubauen.	(1)	(2)	(3)	(4)	(5)
Ich würde mit ‚Presentation’ nicht länger brauchen als mit dem Visual Editor.	(1)	(2)	(3)	(4)	(5)
Ich denke, es ist nicht aufwendiger eine ähnliche Zeitstruktur in einer Presentation–Datei zu begreifen.	(1)	(2)	(3)	(4)	(5)

Figure C.3: The questionnaire (page 3) which the subjects were asked to fill out in the final evaluation of the interactive software prototype after doing the experiment. Before answering section 9) the subjects saw an example for a complex experiment in the Visual Editor which they tried to understand.



10) Hatten Sie Probleme sich im Programm zu Recht zu finden? Wenn ja, wo?

11) Gab es ein Programmelement, dessen Funktion Sie nicht voraussagen konnten oder dessen Funktion anders war als Sie erwartet hätten?

12) Gibt es eine Funktion, die vereinfacht werden könnte oder sollte? Wenn ja, welche?

13) Haben Sie eine Programmfunktion vermisst? Gab es etwas, was Sie gerne tun wollten, aber Sie glaubten, es geht nicht?

14) Haben Sie sonstige Kommentare zum Programm, z.B. zur Stimuluserstellung, zum Trialfenster, zum Zeitmanagement, zur Baumstruktur/Experimentvisualisierung oder zu besonders guten bzw. schlechten Eigenschaften des Programms/ Programmaufbaus?

Figure C.4: The questionnaire (page 4) which the subjects were asked to fill out in the final evaluation of the interactive software prototype after doing the experiment.

Bibliography

Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science Journal*, 126(2):183–235, 1994.

Manfred Amelang and Werner Zielinski. *Psychologische Diagnostik und Intervention*. Springer-Verlag GmbH, 4 edition, 1994.

Hugh Beyer and Karen Holtzblatt. *Contextual Design: Defining Customer-Centered Systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.

Jan Borchers. *A Pattern Approach to Interaction Design*. John Wiley & Sons, Ltd, Chichester, UK, 2001.

Stuart K. Card, Thomas P. Moran, and Allen Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA, 1983.

Cedrus. Superlab 4, 2008. URL `http://www.superlab.com`.

Luca Console, Claudia Picardi, and Daniele Theseider Dupré. Temporal decision trees: Model-based diagnosis of dynamic systems on-board. *Journal of Artificial Intelligence Research*, 19 (2003):469–512, October 2003.

M.F. Costabile, D. Fogli, C. Letondal, P. Mussio, and A. Piccinno. Domain-expert users and their needs of software development. In *The 1st International Conference on Universal Access in Human-Computer Interaction (UAHCI)*, pages 232–236, Crete, Greece, June 2003.

Brian Dorn and Mark Guzdial. Graphic designers who program as informal computer science learners. *The Second International Computing Education Research Workshop, ICER'06*, September 2006.

Jacob Eisenstein and Angel R. Puerta. Adaptation in automated user-interface design. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI) 2000*, pages 74–81, New Orleans, LA, USA, 2000. ACM Press.

B. Horwitz, K. J. Friston, and J. G. Taylor. Neural modeling and functional brain imaging: An overview. *Neural Networks*, 13(8–9):829–846, November 2000.

Oswald Huber. *Das psychologische Experiment: Eine Einführung*. Verlag Hans Huber, 4 edition, 2005.

Jeff Johnson. *GUI Bloopers 2.0: Common User Interface Design Don'ts and Dos*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2 edition, 2007.

Bryan Kolb and Ian Q. Whishaw. *Fundamentals of Human Neuropsychology*. Worth Publishers, 6 edition, 2008.

Jochen Müsseler and Wolfgang Prinz. *Allgemeine Psychologie*. Spektrum Akademischer Verlag, 1 edition, 2002.

Jakob Nielsen. Iterative user interface design. *IEEE Computer*, 26(11):32–41, November 1993. URL `http://www.useit.com/papers/iterative_design/`.

Jakob Nielsen. *Usability Engineering*. Academic Press, San Diego, CA, USA, 1994.

OMG. Unified modeling language: Superstructure - version 2.1.2, 2007. URL `http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF/`.

Karl R. Popper. *Logik der Forschung*. Mohr Siebeck, 10 edition, 1998.

John R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

Jef Raskin. *The Humane Interface - New Directions for Designing Interactive Systems*. Addison-Wesley Professional, 1 edition, 2000.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence, second edition, 2003.

Frank Schneider and Gereon R. Fink. *Funktionelle MRT in Psychiatrie und Neurologie*. Springer-Verlag GmbH, Berlin, Germany, 2007.

Carolyn Snyder. *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

NBS Neurobehavioral Systems. Presentation 12, 2008. URL `http://www.neurobs.com/`.

W3C. Synchronized multimedia integration language (smil 2.1), December 2005. URL `http://www.w3.org/TR/2005/REC-SMIL2-20051213/`.

Philip G. Zimbardo and Richard J. Gerrig. *Psychologie*. Springer-Verlag, 7 edition, 1999.

Index

Typeset July 27, 2008