User Studies of Intelligent Interfaces: Developing and Validating Experimental Designs

Jihie Kim
University of Southern California / Information Sciences Institute
Marina del Rey, CA 90292 USA
+1 310 448 8769
jihie@isi.edu

ABSTRACT
User studies of intelligent interfaces are often very hard and expensive, especially when we evaluate user interactions involved in solving complex problems. Unlike many HCI evaluations where tasks are simpler and target users seem numerous, some of our evaluations require quite an amount of time for the user to think and interact with the system. Although there have been prior work on evaluation methodology and useful heuristics in related areas, when we were designing our evaluations, we needed additional strategies that could guide us more effectively. For example, since we could not find many users who can commit such a long time especially when they are domain experts, in designing our evaluations we needed to ensure that we could gather useful information with a limited number of users. In order to address similar issues in future evaluations, we have compiled a set of strategies for validating and revising our experimental designs. The strategies are general and seem useful for evaluating intelligent interfaces that support complex tasks. We illustrate these strategies using our recent evaluations of Trellis, an intelligent interface that assists users in building structured argumentations.

Categories and Subject Descriptors
H.5.2 [Information Interfaces and Presentations]: User Interfaces – *User interface management systems*; I.2.0 [Computing Methodologies]: Artificial Intelligence

General Terms: Experimentation, Design, Human Factors

Keywords: user evaluations, decision making, argumentation

1. INTRODUCTION
The study of intelligent interfaces increasingly emphasizes the role and the value of user studies and tool evaluations. Related research on user studies can be found in the literature of software engineering where they study various approaches to validate programming environments and the end-to-end software development process [4, 6, 19, 29]. The field of intelligent tutoring systems explores evaluation techniques and measures that are suitable for testing components of tutorial software and learning [16, 23, 24]. There are a lot of studies of human factors and evaluations of human-computer interaction issues [7, 17, 18, 20] and related research on experimental design in psychology [5]. All seem to agree on the difficulty and cost of these studies, as well as on their important benefits. Although the studies of intelligent systems in general haven't been fully exploiting these prior experiences[12], recently there have been some discussions of useful methods and heuristics in designing evaluations for intelligent systems [9] including work on evaluations of intelligent interfaces [11, 22, 25]. In designing experiments of intelligent interfaces, we can draw from the work that is ongoing in the related fields.

We have been developing intelligent interfaces that support complex problem solving tasks such as performing deep analysis on controversial issues or composing complex procedural knowledge [8, 13, 15]. Unlike many HCI evaluations where tasks are simpler, our evaluations require quite an amount of time for the user to think and interact with the system, and for the experimenters to spend time and other resources preparing the experiment (often months) and analyzing the results. In performing evaluations of our interfaces we started from our prior work on developing evaluation methodology for knowledge acquisition tools [27] and other existing aforementioned related work, and adapted the methodology to support evaluations of our interfaces in general.

Although the methodology was very useful, when we were designing and executing our evaluations, we have encountered additional challenges. There were many situations where we had to consider additional factors, such as resource constraints and characteristics of user tasks. For example, unlike other usability tests where the target users seem numerous, we could not find many users who could commit to long testing durations especially when they are domain experts, and we needed to ensure that we could gather useful information with limited resources. Often we needed design our experiments to test more than one hypothesis. Due to potential user variances, we needed to design the tasks and associated instrumentation so that we could collect useful data even when the subjects follow different paths in performing the same tasks.

Although we were using this kind of findings from previous experiments as clues for designing the next one, the validation of our experimental designs and their revisions were done in a rather ad-hoc fashion. There were even times where similar issues reappeared, especially when we started a new experiment for a newly developed interface. Interestingly, although these types of problems seemed common across different evaluations, we had limited guidance on how to validate such experimental designs and how to improve them. It seemed that we needed additional strategies for evaluating a class of intelligent interfaces that support tasks that require much deliberate thinking and long interactions with the system.

We then started to compile a set of general strategies for validating experimental designs based on our experiences. Our goal was to alleviate problems in future evaluations, cutting down some of the unnecessary costs that we had been paying.

The contributions of this paper are twofold. First we present a set of general strategies for validating experimental designs for intelligent interfaces. The strategies have been developed based on our experiences in validating and revising our evaluation designs in the past. Second, we demonstrate our strategies using a series of experimental designs we have developed and executed for a specific system called Trellis, an intelligent interface that assists users to build structured argumentations.

Trellis is a web-based interactive interface for argumentation and decision making. The system allows users to add their observations, viewpoints, and conclusions as they analyze information by making semantic annotations to documents and other on-line resources. Users can intermix free text and structured connectors (i.e. semi-formal representations). Trellis supports collaborative argumentation as well as individual analysis by letting more than one user contribute their annotations to the same analysis.

In evaluating Trellis, we wanted to assess how the annotations created by users help them understand the decision rationale that other users had and how Trellis helps users make high quality decisions. This kind of evaluation seems not in the common areas that HCI usability evaluations focus on but is still in a general class of problems in intelligent interfaces research.

We start with a brief description of the Trellis system. We then show the procedure that we are following for designing and conducting user evaluations, and present our general strategies for validating and revising experimental designs. We describe a series of experiments that we have designed and executed for Trellis where each experiment we attempted could not produce expected results. We show how the strategies guided the validation and the revision of the experiments. Finally we summarize with other lessons learned from these experiments.

2. TRELLIS

Trellis is a mixed-initiative system that helps users annotate information analysis and decision rationale. The Trellis interface helps users enter structured argumentation using semi-formal representation, a combination of formal structures together with free text which is not formalized. That is, the system provides a vocabulary to support semantic annotations for information interdependencies that capture the rationale for decisions as well as traces to the original information sources used. As a mixed initiative system, Trellis reasons with sub-parts that can be formalized, while leaving other parts of the problem to the human and their more thorough understanding of the task.

Figure 1 shows several main components of Trellis: (1) the Main Analysis Composer, (2) the Source Editor, (3) Tree-Trellis and (4) Table-Trellis. They show example analyses from a biography domain. The main analysis composer contains the main topic (Mohammed Atta) and the main statements or (intermediate) conclusions that are reached for the topic. The Main Analysis Composer allows a mixture of arbitrary free text with structure argumentation connectors such as "is elaborated by", "provides background for ", "in contrast with", etc. Such rich annotations are hidden in the collapsed view in Figure 1. The Source Editor allows the user to collect useful information sources and their portions that are relevant to the analyses. The sources and source portions are used as evidences supporting the user entered statements.

Tree-Trellis focuses on collaborative argumentation with potentially contradictory information sources. Unlike the Main Analysis Composer, Tree-Trellis supports only two semantic connectors: pro and con. Due to its simplicity, Tree-Trellis helps users enter argument contributions in a more consistent manner, and let them focus on contradictory views and conflicting information sources. The discussions in Tree-Trellis can be connected to the main statements in the Main Analysis Composer. For example, the discussion of Tree-Trellis in Figure 1 is connected to the statement "2001 Apirl, Atta believed to have met an Iraqi official" in the main analysis composer.

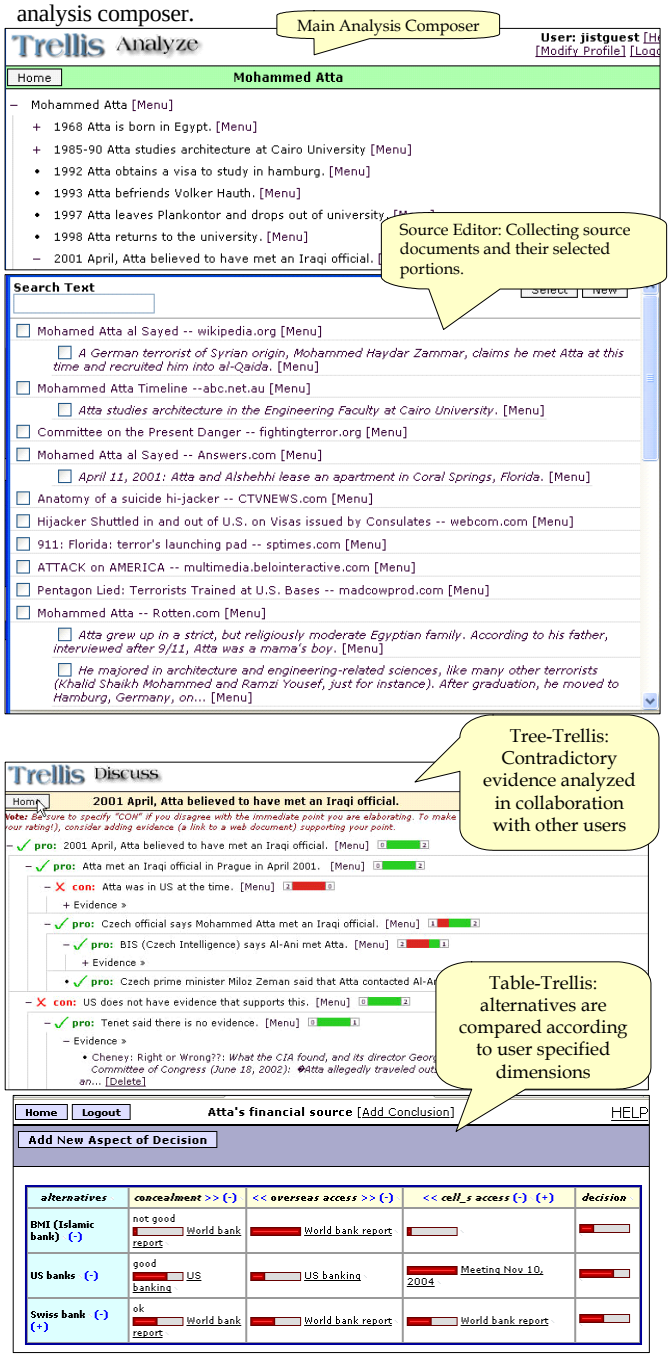


Figure 1. Main components of the Trellis system.

Table-Trellis has been recently added to the system to allow users use table forms to annotate decisions involving multiple alternatives.

It supports a concise presentation of comparisons of alternatives. The development of these components was motivated by an analysis of collections of arguments and statements spanning a variety of topics from web users. More details of the tools and their characteristics can be found in [8].

We wanted to evaluate the benefit of these capabilities including how the interface features in Trellis assist users in generating argumentation or modify existing argumentation based on new information sources.

3. EXPERIMENT PROCEDURE

1. State general claims and specific hypotheses -- what is to be tested
2. Determine the set of experiments to be carried out -- what experiments will test what hypotheses
3. Design the experimental setup
   (a) Determine the task to be performed -- what kinds of actions to what kinds of features
   (b) Choose type of users that will be involved -- what kind of background and skills
   (c) Design the experiment procedure -- what will the subjects be told to do at each point
4. Determine data collection needs -- what will be recorded
5. Perform experiment
6. Analyze results -- what results are worth reporting
7. Assess evidence for the hypotheses and claims -- what did we learn from the experiment

Figure 2. Steps to design and conduct experiments.

Figure 2 shows the basic steps we are following in conducting evaluations of intelligent interfaces. We adapted the steps from the methodology that we have developed for knowledge acquisition tools [27]. The procedure is not strictly linear and there could be iterations and backtracking across the steps due to intermediate assessment and interactions between evaluation constraints and associated decisions.

The first step in the design of our evaluations is to state the main claims. Claims refer to general capabilities of benefits of a system. Given these claims, we could state specific and measurable hypotheses to be proved or disproved with experiments that are feasible given our resources and constraints. In practice, many hypotheses are hard to evaluate because they imply experiments that may be unfeasible due to lack of time and other resources. In order to show the benefits of a tool or technology, a useful way to design an experiment is to perform a comparison with some baseline tool. That is, in determining the set of experiments to be carried out, we often use *tool ablation* experiments, where the baseline tool results from eliminating some capability of the tool. The group of subjects that is given the ablated tool serves as the *control* group. We found these experiments to be the useful kind to test claims about interface features. Due to limited resource constraints in terms of the number of users available we often performed *within-subject experiment* where each user performs two different but comparable set of tasks.

Once we have determined the hypotheses and the kind of experiment to be carried out, we can plan on the details of the experiment including the subjects, tasks and the specifics of the experimental procedure. The experimental procedure includes, for example, what information will be given to the subjects and in what format, what kind of interaction can the subjects have with the experimenters during the tests (if any), how many tasks will be given to each subject and in what order, and an indication of the success criteria for the subjects so they know when they have finished the task.

The kind of data collected during the experiment may be determined by tool instrumentation that is feasible as well as the hypotheses. We often use automatic logs of the interface features that are used by the user, time to complete a task, resulting changes in the status of the system, and detailed notes taken manually. Questionnaires at the beginning and end of the experiments seem useful in assessing subject background and performing additional qualitative assessment. Videotaping can be used if needed but we end up not using it in our data analysis due to the cost of processing the tapes. Intrusive ways of recording data should be avoided.

After the experiment is carried out according to the pre-determined procedure, the raw data should be carefully examined and normalized in order to support assessment of whether the hypotheses and claims are proved or disproved. Sometimes data relevant to the hypotheses have to be isolated due to uncontrolled factors, which often require detailed analysis of gathered data.

It is very useful to run a pre-test using a preliminary version of the experimental setup so that the design of the overall experiment can be debugged and validated.

4. STRATEGIES FOR VALIDATING EXPERIMENTAL DESIGNS

Over the last several years we have performed a series of user evaluations of our tools that support complex problem solving. We had various types of users, ranging from people from our group to end users who do not have computer science background such as military officers and biologists[3, 13, 14]. In many cases, we rarely succeeded at our first attempt. The evaluations often did not go as we expected and we needed to go back to re-design and re-execute the experiment, resulting in cycles of design, execution, and diagnosis. These experiences have allowed us to develop a set of strategies for validating experimental designs and improving them. The strategies can be used to analyze results from pre-tests or actual evaluations but can also guide the steps in Figure 2. The following presents the strategies that we have developed.

- S1: Check the experiments determined given the resource constraints. Check if any of the chosen hypotheses could or could not be tested.
- S2: Check the tasks assigned to the user: If proving or disproving the hypotheses involves evaluating some tool features and the tasks did not clearly support evaluations of the features, modify the tasks and/or choose different domains.
  - Ensure that for each path the user could follow in performing the tasks, there are enough user actions that involve uses of the tool features. (e.g. Avoid use of too simple tasks)
  - If the tasks did not allow collection of the data needed, modify the tasks to support the data collection (e.g.

Break down complex tasks into smaller tasks to allow finer grained analysis)
- If there were undesirable interactions between the tasks, modify the tasks or change the execution procedure to remove the interactions (if possible).
- If the characteristics of the given domain did not support evaluation of the tool features, pick a different domain that allows it.

- S3: Check the users involved: If there were assumptions made for the users and the results were inconsistent with the assumption, use the training phase and questionnaire to identify differences among the subjects. Ensure that the users are motivated to perform the complex tasks.
- S4: Check the tool features:
  - For tool ablation tests or tool comparisons, identify the features to be compared or removed and ensure that the ablated version (or the baseline system) satisfy the requirements for the evaluation. (e.g. The features should be clearly removed in the ablated version.)
  - If the selected tool features were not appropriate for testing the hypotheses, identify/develop better features. (e.g. data from uses of selected tool features did not provide enough evidence.)
- S5: Check metrics and data collection: If the data collected did not support testing the hypotheses, improve the tool instrumentation, modify the experiment procedure, or modify the tasks to allow better (fine grained) data collection. Ensure the metrics should allow identification of the steps that are relevant and irrelevant to the hypothesis.
- S6: Check experiment procedure: If the procedure didn't go as expected or users didn't follow the procedure as expected, modify the steps or the materials prepared. Ensure that the instructions are appropriate for the users to understand how to perform complex tasks.
- S7: If the above strategies could not be followed, revise claims and hypotheses based on the constraints of the evaluation components.

Figure 3. Strategies for validating and revising experimental designs.

Given an experimental design, we check the key components of the design including the experiments that were determined to test the hypotheses, user tasks, users involved, tool features, data collection and the experiment procedure, assessing whether any of them need to be refined in order to produce more useful results.

Due to the cost of performing experiments, we often test more than one hypothesis with the same experiments. In checking the set of determined experiments, we examine whether the chosen hypotheses can be tested with them. For example, in one of the experiments, we wanted to know how much training is needed to learn to use our tools in addition to testing how useful our tools are. This testing was done due to the complexity of the tasks and the interactions associated with them. We have used the same experiment but needed to ensure that the users acquire a comparable level of skills before they move on to the actual tasks. We needed to re-organize the training procedure in order to measure their levels of understanding [15].

In designing or selecting the tasks assigned to the users, we often made mistakes of using too simple tasks that do not effectively allow testing of the tool features that we are interested in. This seems obvious but sometimes we had more than one feature that assist various aspects of user tasks, we needed to thoroughly check what subtasks are supported by what features. This becomes harder when there are alternative paths the subjects could follow to accomplish the tasks. We needed to ensure that the alternative paths are comparable and to include enough episodes that require uses of the tool features. In some other cases, we had problems in the way we were using complex tasks. When we gave them as one single task and the user completed almost but not all of it, we could not measure how many (sub) tasks the user could complete since a mistake in one sub-task led to problems in starting the next sub-tasks. In order to collect finer grained data about how many tasks the subjects could complete, we had to divide the task into smaller subtasks and assigned them individually [14].

The domains should be selected carefully so that they support the testing of the hypothesis. We often ignore the details of the given domain but the characteristics of the domain sometimes can determine the feasibility of the test. For example, we could not determine the correctness of the user entered information due to the nature of the domain as described in Section 5. We also need to remove any undesirable interactions among the tasks such as prior tasks providing undesirable hints to the following ones.

It would be preferable to use many subjects for the experiment. However, most experiments we conducted were very expensive and we needed to be careful in deciding who should participate. For example, the tasks we had often required a long period of time (such as collaborative argumentation with a set of information sources). If the studies have some constraints or assumptions on the users, it is desirable to check thoroughly whether the constraints were satisfied before investing expensive resources. Unfortunately, unlike in other studies of interfaces where users seem numerous, we didn't have many choices since it was hard to find subjects who can commit to long testing durations, especially when they are subject matter experts. However, questionnaires and analysis of the training phase seemed helpful in identifying differences among the subjects. We could relate them to the differences in the resulting data. For example, we could group the subjects based on the performance in the training phase and analyze the results based on the grouping. An additional important strategy is to ensure that the users are motivated to perform complex tasks. In the past, some subjects performed the assigned tasks significantly differently than others even if they had similar background and skills [14].

Results from tool ablation study or comparison with a baseline tool may provide clues of whether the selected tool features actually make a difference. If there was no significant difference between the two compared systems, we may consider improving the features or testing different features. However, before making such judgment we should ensure that the baseline tools that we were using satisfy the test requirements. For example, if the two systems rely on different editors that are not related to the claim and we were comparing the overall performance, we may not know how the tool features actually contribute to the difference. If we were using an ablated version of an interface, the features we want to test should be correctly removed.

We design metrics and data collection based on the given hypotheses. However, it is always desirable to use fine grained measures and tool instrumentations especially when there are complex interactions among user actions. Although it is not always possible, fine grained measures have helped us distinguish the steps that are relevant and irrelevant to the claims [26]. They also have helped us validate experiments according to our strategies.

The experiment procedure needs to be examined to check whether the planned steps were followed as expected. Besides the understandability of the instructions, we needed to know whether there are any undesirable interactions among the steps including instructions, training, sequence of subtasks assigned, questionnaire, etc. For example, although we want to let the users learn the tool features during the training phase, we don't want to expose unnecessary hints. If there were undesirable interactions among the assigned tasks, we may need to change the execution procedure that can interrupt such interactions (e.g. changes of the ordering). If such interruption is not possible, we need to modify the tasks to remove such interactions. In scheduling the experiment procedure we should allocate enough time to finish the tasks (if time is not a controlled factor) and get feedback from the users.

If none of these components seem invalid, we can consider revisiting the initial claims and hypotheses that we need to test. (We could consider modifying the hypotheses earlier depending on how strongly the findings point to faulty claims/hypotheses.) In this case, we need to consider constraints of the other components and their interactions with the new hypotheses. In fact there are tight dependencies among the components as shown in Figure 4.

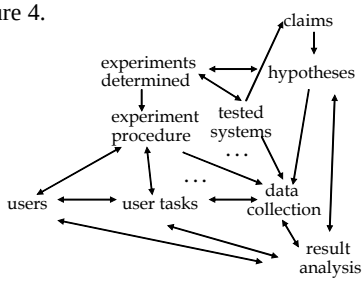


Figure 4. Tight dependencies among evaluation components.

5. DESIGNING USER EVALUATIONS FOR TRELLIS

This section shows a sequence of user evaluations that we have performed for the Trellis system. In particular we illustrate some of our validation strategies by describing how our experimental designs were assessed and modified.

5.1. Experimental Design #1

Our claims included that the structured argumentation of Trellis helps users build new argument or modify existing argument. We decided to test the Tree-Trellis features that support pro/con argumentation first. This was motivated by the need for the tools that support various cooperative works including collaborative analysis [2]. The key features of the Tree-Trellis interface include its support of pro/con annotations of conflicting evidences and argument contributions (i.e. statements) and hierarchical argumentation structure that presents intermediate hypotheses and decision rationale.

We wanted to know whether the annotations were useful for collaborative decision making where more than one user contribute to the decision. In particular we wanted to evaluate whether these annotations assist users in modifying existing decisions (possibly made by other users) better than traditional reports. We formed the following hypothesis to support the evaluation.

Hypothesis I: pro/con annotations of decisions with Tree-Trellis help users modify existing decisions (possibly made by other person) more correctly over traditional report because 1) pro/con annotations expose rationale for the decisions, intermediate hypotheses and supporting sources and 2) pro/con annotations help users understand the views other users had.

In order to collect data comparable across users and tasks, we have used a controlled experiment. We have instrumented the tools to collect and compare data about how users would perform these tasks under two conditions: argumentation with the Trellis annotation and one without the annotations. We designed an ablated version of the argumentation that present the same decision but without pro/con annotations. In the ablated version, the same set of evidences was presented to the user but they were given as a list without pro/con annotations and hierarchical structure.

We have used the domain of composing biographies where users argue about activities of a person, whether a person performed an action at certain time. The analyses could base on some conflicting evidences. The pro/con annotations that we have used are similar to the annotations in the Tree-Trellis interface shown in Figure 1.

Since we were interested in incremental changes in the analysis during collaborative analyses, the tasks were broken down into a sequence of subtasks. For each subtask, the subjects were given a set of new evidences and asked to modify the existing decision based on the new evidences. In order to simulate the actual work environment more closely, we have added some 'intrusive' steps between the subtasks so that the task relevant information would be forced out of their short-term memory before they start the next task. For example, we have asked them to take a lunch break, perform very different analysis tasks such as analyzing a news article, talk about their recent vacation, etc. This was done based on some discussions with people who work on psychology experiments.

Given a set of evidences, the subjects entered additional statements and made associated changes in the ratings of the statements including the top-level statements (i.e. conclusions). For a statement, supporting evidences and its pro statements would increase its rating and opposing evidences and its con statements would decrease it. Tree-Trellis allows users to either rate or vote on statements. For this evaluation, we have used the rating function that allows the users enter or modify the ratings of any statements.

In order to test the hypothesis we attempted to measure the correctness of the modification, i.e., whether the subjects changed the ratings *correctly*. In order to understand what things users found hard and difficult to do with the tool or why certain action was taken or not taken, we collected additional information during the experiment. We asked users to voice what they thinking and they were doing and recorded them in transcripts. The total time spent was less important for our evaluation.

Before we start actual evaluation with subjects who are not familiar with the tools, we performed a pre-test with people in our group.

Analyzing Experimental Design # 1

As we finished the pre-test for this evaluation, we began to notice there are some problems in our experiment. We started checking our experiment with our strategies. Strategies *S1, S2, and S5* helped us realize that with given domain, our hypothesis could not be proven with the collected data. In the domain (argumentation about a person's activities) where the user had to combine a set of conflicting evidences to determine the truth of an event, it was hard to determine whether the user modified the decisions correctly. For example, when the user was analyzing whether a person was present at a location at a certain time with some evidences (e.g. there is a witness), there could be unpredictable situations (e.g. another person is disguising to be the person) and we could not correctly determine or assess the possibility of whether the person was actually at the location at the time. Following the strategies, we needed to use a domain where the correctness can be determined and create a new set of user tasks accordingly.

In modifying the experimental design, strategy S6 helped us revise experiment procedure based on the additional findings from our transcript and questionnaire. The intrusive steps that we have added to our experiment procedure didn't help the subjects forget task-relevant information. They have told us that they were able to perform the subtasks as if they were doing them continuously without any interruptions. We decided to not to use the tasks involving incremental changes in our experiment.

Strategy S4 helped us realize a problem with the baseline system since it is not very comparable with traditional reports as we initially planned. We developed a new version of the tool that is designed based on typical forms of the traditional reports that we have collected from several analysts. The reports start with the topic and some conclusions, and the supporting details or justifications are organized in some paragraphs. Usually information that provides alternative views is not explicitly presented in the details. Related information sources are listed at the end as references. Other interface features including the editors that are used to enter new statements or change the ratings remained the same.

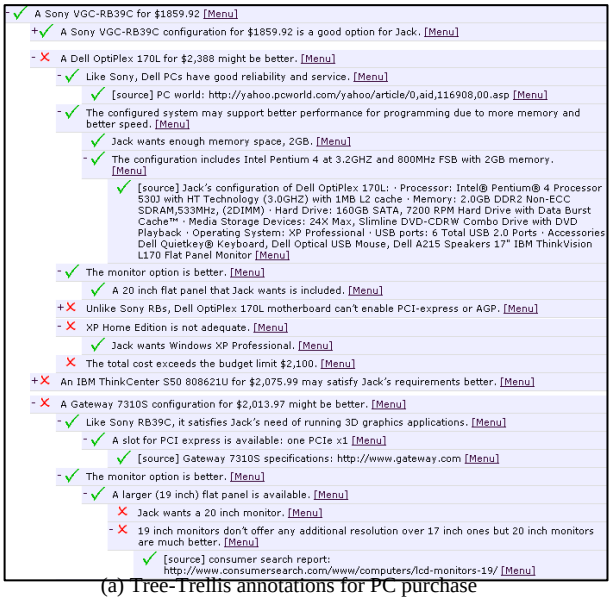
5.2 Experimental Design # 2

For the new design, we have chosen a domain of computer purchase. The subjects were asked to find good computer configurations for a person that are better than the options the person had looked at. The pro/con annotations present strengths and weaknesses of computer configurations that were considered. The annotations expose the constraints on computer features and rationale for choosing the most likely one. They also point potential weakness (cons) of an intermediate decision and how the existing decision could be updated.

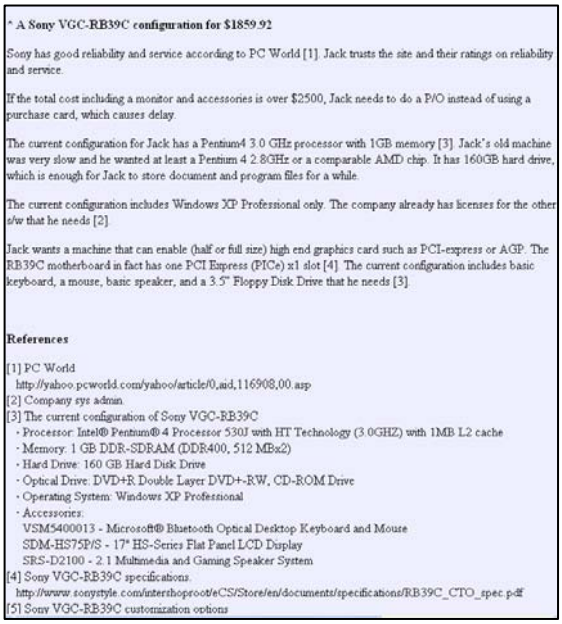
The correctness of the user entered modification was measured by judging whether the new option (i.e. a computer configuration) selected by the subject satisfies all the decision constraints that were considered in analyzing the other options.

Since we were interested in the benefit of pro/con annotations over traditional reports, the baseline system took a format of traditional reports as described above. It had the details including the justification of the decision but the details were organized in a set of paragraphs and references. Figure 5 shows some part of the annotations. The hypothesis remained the same.

Hypothesis II: Same as Hypothesis I.



(a) Tree-Trellis annotations for PC purchase



*** A Sony VGC-RB39C configuration for $1859.92**

Sony has good reliability and service according to PC World [1]. Jack trusts the site and their ratings on reliability and service.

If the total cost including a monitor and accessories is over $2500, Jack needs to do a P/O instead of using a purchase card, which causes delay.

The current configuration for Jack has a Pentium4 3.0 GHz processor with 1GB memory [3]. Jack's old machine was very slow and he wanted at least a Pentium 4 2.8GHz or a comparable AMD chip. It has 160GB hard drive, which is enough for Jack to store document and program files for a while.

The current configuration includes Windows XP Professional only. The company already has licenses for the other s/w that he needs [2].

Jack wants a machine that can enable (half or full size) high end graphics card such as PCI-express or AGP. The RB39C motherboard in fact has one PCI Express (PICe) x1 slot [4]. The current configuration includes basic keyboard, a mouse, basic speaker, and a 3.5" Floppy Disk Drive that he needs [3].

References

[1] PC World
http://yahoo.pcworld.com/yahoo/article/0,aid,116908,00.asp
[2] Company sys admin.
[3] The current configuration of Sony VGC-RB39C
- Processor: Intel® Pentium® 4 Processor 530J with HT Technology (3.0GHZ) with 1MB L2 cache
- Memory: 1 GB DDR-SDRAM (DDR400, 512 MBx2)
- Hard Drive: 160 GB Hard Disk Drive
- Optical Drive: DVD+R Double Layer DVD+-RW, CD-ROM Drive
- Operating System: Windows XP Professional
- Accessories:
  VSM5400013 - Microsoft® Bluetooth Optical Desktop Keyboard and Mouse
  SDM-HS75P/S - 17" HS-Series Flat Panel LCD Display
  SRS-D2100 - 2.1 Multimedia and Gaming Speaker System

[4] Sony VGC-RB39C specifications.
http://www.sonystyle.com/intershoproot/eCS/Store/en/documents/specifications/RB39C_CTO_spec.pdf
[5] Sony VGC-RB39C customization options

(b) Baseline version

Figure 5. Example pro/con annotations used.

Before we start the actual evaluation, we have checked the feasibility of the tasks by performing a pre-test with some students in our group. The pre-test showed that the earlier problems with the Design #1 have disappeared. We have decided to move to an actual test.

For the actual evaluation, we have used four subjects who were not familiar with our projects. We have performed within-subject experiment with both Tree-Trellis and the baseline version as described earlier.

Analyzing Experimental Design # 2

With the baseline tool (without the pro/con annotations), none of the subjects were able to produce a correct result. With pro/con annotation two of the subjects produced correct results that satisfy all the decision constraints. We have found that two 'hasty' users overlooked one decision constraint each. One user didn't pay attention to the operating systems constraint on a Windows system. The subject said that he is familiar with Mac systems that have only one operating system. The other subject didn't read all the pro/con annotations that are provided and missed a constraint on selecting redundant power options for computer servers.

There were some additional findings about the tool features. Subjects commented that with the pro/con annotations, the supporting sources are located close to the statements, which helps them find relevant information more easily. Also some subjects also commented that pro/con annotations make tracking decision constraints easier than traditional reports.

We started to check the experiment against our strategies. Although there were some useful findings, the results didn't fully satisfy our expectation that users should produce correct results with Tree-Trellis since only the half of the users modified decisions correctly with the pro/con annotations. Following Strategy S4, we have decided to explore another feature that allows presentation of decision constraints -- Table based annotations. Table-Trellis supports a concise presentation of comparisons involving multiple alternatives. Following Strategy S3 and S6,we also improved the experiment procedure to ensure that all the users read all the information that is given to them. In the new instructions, we emphasized this in order to reduce any potential variances in reading the information prov ided.

5.3 Experimental Design # 3

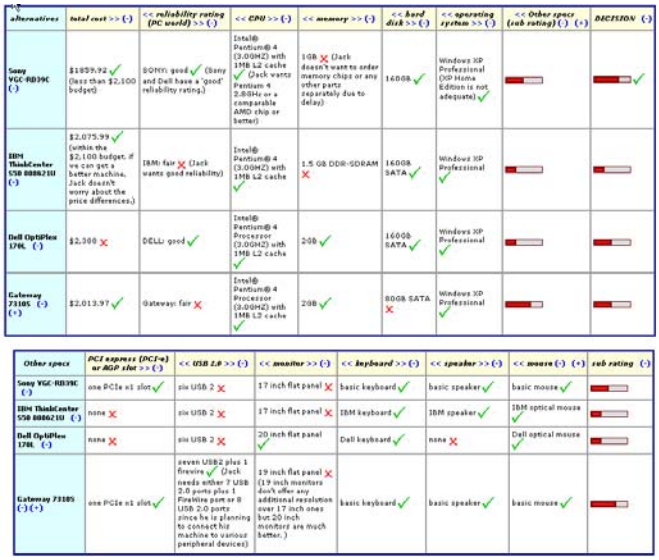


Figure 6. Example pro/con annotations used for Table-Trellis.

In order to assess the benefit of concise table-based annotations, we have decided to form the following hypothesis.

Hypothesis III: Table-based annotations help users update decisions in light of new options more *correctly* and *efficiently* over Tree-Trellis because Table-based annotations present all the decision constraints used in comparing different options more concisely.

Analyzing Experimental Design # 3

Table 1 shows some of the results from the evaluation. We have found that the users finish the tasks 40% faster with the table-based annotations. The completion time with pro/con annotations were similar to the results with Tree-Trellis in the prior test (with Design #2). All of the users produced the correct results with Table-Trellis. With Tree-Trellis two users produced incorrect results. Although we improved the experiment procedure to ensure that all the subjects read the presented information, some users still seemed to have difficulty in going through rather wordy descriptions of decision constraints in Tree-Trellis.

	Tree trellis	Table Trellis
Avg time to read existing annotations	15.5	9.25
Avg time to complete the tasks	35.75	21.5
Number of incorrect configurations	2	0
Number of features violated	4	0

Table 1: Some results from evaluations of Table-Trellis

When we checked the experiment with our strategies, all of the experiment components seem valid and the data produced was supporting the hypothesis. Although we need more data (and more users) to fully prove the hypothesis, the design of the experiment seemed valid.

As in the other evaluations, the detailed transcript provided additional information about the evaluation and the tool features. Although table-based annotations make the tasks easier and faster, users commented that simple values in the tables are less informative than the statements in pro/con annotations and they often needed to infer the given situation in order to understand the decision constraints well. Although the tasks that we have used did not require knowing the details of the situation, in order to support a broad range of tasks, we may need to make such information available in some way.

4.4 Summary of Evaluations and Results

Our validation strategies seem helpful in finding potential problems in our evaluation designs and revising them. While we were going through three different designs and two pre-tests, we were able detect most of the problems early on and collect useful information from the experiment.

Although we need more data, the information we have collected tells us that Tree-Trellis helps users understand and modify existing decisions but Table-based annotations make the tasks more efficient and more correct. However we expect that for the tasks that require deeper understanding of the decisions, the annotations in Tree-Trellis can be helpful.

6. RELATED WORK

We can find relevant research in other fields including software engineering, HCI and tutoring systems. In software engineering, empirical evaluations involve testing of many different aspects and issues in the software development process including language, development environment, reuse, quality, software management, etc [4, 19, 21]. User studies are only a part of these

tests since many of these steps do not heavily rely on user involvement. Although our work does not focus on the end-to-end process of solving complex problems, including identifying problems, testing the feasibility, performing an initial subtask, keeping track of the outcome, etc., we could consider adopting similar techniques for the tools that support the relevant parts of the end-to-end process.

In evaluating intelligent tutoring systems, the tradeoffs regarding the benefits and suitability of different evaluation approaches are analyzed [23, 24]. Although formal studies are desirable in some cases, informal studies (such as Wizard of Oz techniques) seem common and sometimes preferred due to the factors relevant to the learning process rather than systems evaluations [28].

The studies of HCI [1, 17, 18] share many issues that we have and we draw from their work. Some of the work provides strategies for validating the experiment such as selection of users and instrumentation, how statistical conclusions can be made from data, etc. [10]. However, in contrast to our work, their tasks tend to be simpler and the target users seem numerous. In order to guide evaluations of our interfaces that support more complex tasks with a small number of subjects, we needed to develop additional strategies.

We are also looking at additional useful methods and heuristics in evaluating intelligent interfaces [11, 22, 25]. Our strategies share some of the issues (e.g. task design, user variances, metrics) and complement their approaches by providing additional strategies that can be more effective for evaluations of interfaces that support complex tasks.

7. SUMMARY

This paper presents a set of general strategies that we have developed for validating and revising experimental designs. The strategies can be used in combination with other general methodologies and seem useful for detecting problems in evaluation designs early on. Running one or more pre-tests with these strategies before finalizing the experimental design seemed desirable. Since they were built based on our past mistakes and lessons learned, they point to issues we often overlook and may cut down some of the unnecessary costs.

8. REFERENCES

[1] Baecker, R., Grudin, J., Buxton, W. and Greenberg, S. *Readings in human-computer interaction: Toward the year 2000*. Morgan Kaufmann, 1995.

[2] Baecker, R.M. *Readings in Groupware and Computer-Supported Cooperative Work: Assisting Human-Human Collaboration.* Morgan-Kaufman, San Francisco, CA, 1993.

[3] Barker, K.et.al., A Knowledge Acquisition Tool for Course of Action Analysis. in *IAAI-2003*, (2003).

[4] Basili, V., Selby, R.W. and Hutchens, D.H. Experimentation in Software Engineering. *IEEE Transactions in Software Engineering, SE-12* (7).1986

[5] Breakwell, G.M., Hammond, S. and Fife-Schaw, C. *Research Methods in Psychology*, London, 2000.

[6] Brown, P.S. and Gould, J.D. An Experimental Study of People Creating Spreadsheets. *ACM Transactions on Office Information SYstems, 5* (3). 258-272.1987

[7] Byrne, M.D. and Gray, W.D. Returning human factors to an engineering discipline: Expanding the science base through a new generation of quantitative methods - Preface to the special section. *Human Factors, 45* (1).2003

[8] Chklovski, T., Ratnakar, V. and Gil, Y., User Interfaces with Semi-Formal Representations: a Study of Designing Argumentation Structures. in *Proceedings of IUI-2005*.

[9] Cohen, P.R. Empirical Methods for Artificial Intelligence.1995

[10] Gray, W.D. and Salzman, M.C. Damaged Merchandize? A Review of Experiments That Compare Usability Evaluation Methods. *Human Computer Interaction, 13*. 203-261.1998

[11] Hook, K., Tutorial: Designing and Evaluating Intelligent User Interfaces. in *IUI-1998*, (1998).

[12] Howe, A.E. and Cohen, P.R. How Evaluation Guides AI Research: The Message Still Counts More than the Medium. *AI Magazine, 9* (4).1988

[13] Kim, J. and Blythe, J., Supporting Plan Authoring and Analysis. in *IUI-2003*, (2003), 109-116.

[14] Kim, J. and Gil, Y., Acquiring Problem-Solving Knowledge from End Users: Putting Interdependency Models to the Test. in *Proceedings of AAAI-2000*, (2000).

[15] Kim, J. and Gil, Y., User Studies of an Interdependency-Based Interface for Problem-Solving Knowledge. in *IUI-2000*, (2000).

[16] Mark, M.A. and Greer, J.E. Evaluation methodologies for intelligent tutoring systems. *Journal of Artificial Intelligence and Education, 4* (2/3). 129-153.1993

[17] Nielsen, J. *Usability Engineering*. Academic Press, 1993.

[18] Olson, G. and Moran, T. Special issue on experimental comparisons of usability evaluation methods. *Human-Computer Interaction, 13*.199

[19] Pfleerger, S.L. Experimental design and analysis in software engineering. *Annals of Software Engineering, 1*. 219-253.1995

[20] Preece, J. and Rombach, H.D. A Taxonomy for Combining Software Engineering and Human-Computer Interaction Measurement Approaches: Towards a Common Framework. *International Journal of Human-Computer Studies, 41*.1994

[21] Rombach, H.D., Basili, V.R. and Selby, R.W. (eds.). *Proceedings of the International Workshop on Experimental Software Engineering Issues: Critical Assessment and Future Directions*, 1992.

[22] Scholtz, J. Evaluation Methods for Human-System Performance of Intelligent Systems. NIST ed. *PerMIS 02*, Gaithersburg, MD, 2002.

[23] Self, J. Special issue on evaluation. *Journal of Articial Intelligence in Education, 4* (2/3). 129-413.1993

[24] Shute, V.J. and W., R.J. Principles for evaluating intelligent tutoring systems. *Journal of Artificial Intelligence and Education, 4* (2/3). 245-271.1993

[25] St. Amant, R. and Dullburg, M., An Experiment with Navigation and Intelligent Assistant. in *IUI-1998*, (1998).

[26] Tallis, M. A Script-Based Approach to Modifying Knowledge-Based Systems, University of Southern California, 2000.

[27] Tallis, M., Kim, J. and Gil, Y. User Studies Knowledge Acquisition Tools: Methodology and Lessons Learned. *Journal of Experimental and Theoretical Artificial Intelligence, 13* (4).2001

[28] Twidale, M. Redressing the Balance: The Advantages of Informal Evaluation Techniques for Intelligent Learning Environments. *Journal of Articial Intelligence in Education, 4* (2/3). 155-178

[29] Zelkowitz, M.V. and Wallace, D. Experimental Models for Validating Computer Technology. *IEEE computer, 31* (5).1998