Facilitating and Automating Empirical Evaluation

Laurian Hobby, John Booker, D. Scott McCrickard, C. M. Chewar, Jason Zietz
Center for HCI and Department of Computer Science
Virginia Polytechnic Institute and State University (Virginia Tech)
Blacksburg, VA. 24061-0106
(540) 231-7409

ABSTRACT

Through the automation of empirical evaluation we hope to alleviate evaluation problems encountered by software designers who are relatively new to the process. Barriers to good empirical evaluation include the tedium of setting up a new test for each project, as well as the time and expertise needed to set up a quality test. We hope to make the evaluation process more accessible to a wider variety of software designers by reducing the time and effort required for evaluation through the use of a wizard-like system that does not require expertise in evaluation techniques. Implementation is accomplished by utilizing a library of design knowledge in the form of claims to focus the evaluations. User tests were performed to evaluate receptiveness to the software tool as well at the performance of the underlying methods. Results were positive and provide a justification for further research into this area as well as exposing problem areas for improvement.

Categories and Subject Descriptors

D.2.2 [**Design Tools and Techniques**]: Evolutionary Prototyping

General Terms

Design, Experimentation, HCI, Human Factors, Verification

Keywords

Critical parameters, design, reuse, interaction.

1. INTRODUCTION

Empirical evaluation, the scientific study of users working with and evaluating a computer system, is naturally a challenging problem for interface design in software engineering in three different yet related ways [11]. The first reason is that it requires the designer to spend a cumbersome amount of time evaluating and analyzing data. It is not hard to imagine that a designer could spend weeks, if not months, evaluating two dozen hours of gathered recording. To extend this problem, when creating software the designer may have to implement multiple iterations of design where the system is tested by a prototypical user. The sheer amount of data produced and the amount of time to analyze this work is an immense process. Also, the setup and execution of creating an empirical evaluation is a unwieldy process.

The second way that empirical evaluation is a challenging problem is that it is a hard process to learn; it requires software engineers to become expert in human-computer interaction (HCI) [9]. Whole college level classes are dedicated to attempting to teach the method of empirical evaluation. However, it is not until the designer is actually attempting to evaluate his software that he finally starts to work out the intricacies of the process. Empirical evaluation, like many things, is a skill and it hard to learn from a book. This can cause the designers to become distracted from the end goal: to support, with data, that the hypothesis was or was not rejected.

The final way that empirical evaluation is challenging is the lack of reusability. The HCI community has been chasing the elusive idea of reusability for decades because it would reduce time and effort for future designs. However, currently there is a critical loss of design data due to the lack of a framework to support reuse. [12]. In the particular case of empirical evaluation, the mass of data that is accumulated in this process is essentially discarded and never used again. If there was a way to leverage the reuse of even a small part of this phase in design, there could be a significant reduction to the time that empirical evaluation would take.

To address these three problems we propose an automated empirical evaluation design environment that would reduce the time needed to complete a study, help novice designers learn the process, and support reuse. We will discuss an application of empirical evaluation and our proposed method of assessment.

2. BACKGROUND

2.1 Notification Systems and LINK-UP

In order to facilitate our goal of supporting an automated empirical evaluating environment we narrowed the concentration of our design to the area of Notification Systems in HCI. Notification systems are a specific kind of software that mediates the idea of multi tasking interfaces while working. Designers employ notification systems because they support reacting to and the comprehension of time sensitive information in a way that is not overly interruptive [6]. For example, the user may have one primary task, but will have multiple pieces of software running in the background. To keep the user up to date on the status of secondary programs, notification systems are implemented to supply the user with incident information. Typical examples are artifacts such as news tickers, automobile dashboard displays, context-sensitive help agents, or blinking icons in the control bar.

LINK-UP (Leveraging Integrated Notification Knowledge with Usability Parameters) is a system that supports all phases of design for general software development by being implemented specifically for notification systems [4]. It allows the designers to have a clear view of the envisioned system while being able to consider the tradeoffs of using particular artifacts. The designer is guided through Scenario Based Design [2] and creates prototypical user scenarios about interaction with the system. Then, the designer creates claims on what are the upsides and downsides for implementing a specific artifact. An example of a claim would be the following:

"Use of ticker-tape text-based animation to display news headlines in a small desktop window: Preserves user focus on a primary task, while allowing long term awareness BUT is not suitable for rapid recognition of and reaction to urgent information."

The designer then uses claims to evaluate the tradeoff as he or she moves through the phases of LINK-UP. The beginning phases of LINK-UP are requirements engineering (where designers elicit the needs from stakeholders for a system image using generated tools) and participatory negotiation (where designers and stakeholders gather to negotiate about needed features and artifacts for the proposed system). The designer then prototypes their design and moves onto the later stages of LINK-UP that are analytic evaluation [6] and automated empirical evaluation. From here design knowledge is then transferred to a reuse library that stores design information that is linked to design claims like the above example.

2.2 Critical Parameters: IRC

Critical parameters are quantitative performance metrics that measure not only a raw attribute, but also measure a system's ability to serve a purpose [8]. How they differ from regular parameter is that they go beyond just a "measure of goodness". The only advice one can retrieve from those types of parameters is what "you need more of." On the other hand, critical parameters are set up so that you have a target value in your design specifications, and you know not only if you have too little of a certain parameter, but also too much. For example, the amount of interruption desired in a notification system varies: when there is a highly critical piece of information that the user needs then a high interruption is desire. Since critical parameters are structured in such a way that you want to match a certain set of "design parameters," it is important that designers and users alike can understand the criteria set forth by a certain parameter. Therefore, critical parameters are important to facilitate communication between the groups of stakeholders for more efficient work.

IRC stands for interruption, reaction, and comprehension. These are the three critical parameters for notification systems developed by McCrickard, et al [5] that we based our empirical system upon.

2.2.1 Interruption

In a dual task computer environment, attention is divided between a primary and secondary task. The interruption parameters capture how much of this attention is required by the secondary task. Whenever a notification system causes the user to shift her focus to the notification system, this is considered interruption. The best way to record this is to measure the effect a notification system has on a user's "primary task performance." Low interruption would be something that does not interfere with the primary task in any noticeable way, such as an icon that changes color once to indicate a new message, but does not blink.

2.2.2 Reaction

In addition to having to look at an interface to discover information, users may be prompted to interact with it. Clicking to see additional information, sending a quick email, or adding an item to a to-do list would be considered reaction. A user goal of low reaction means users desire little direct interaction, such as running a stock ticker for casual observation, not active trading.

2.2.3 Comprehension

Comprehension is the final parameter that measures how much information the user wants to absorb and retain. Low comprehension would indicate that the user did not want to retain much information—for example, if an email notification system used a simple color change in the system tray, prompting reaction but delivering little comprehension.

2.3 Empirical Evaluation and LINK-UP

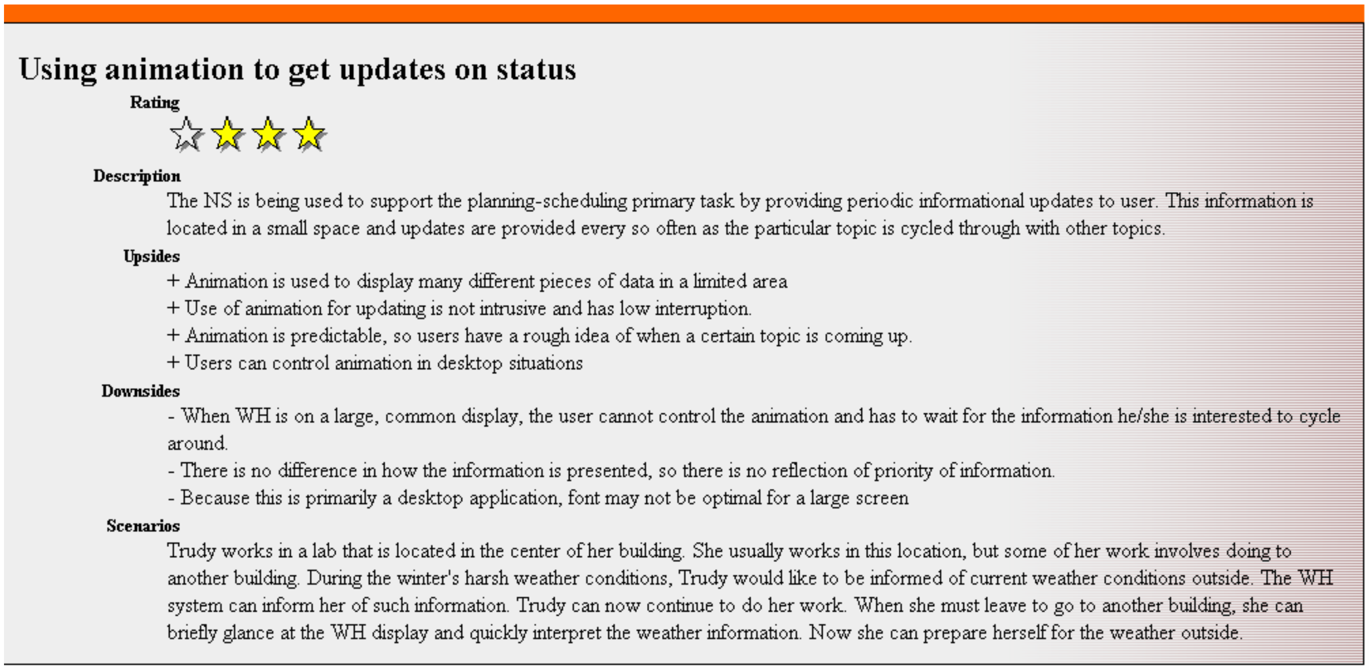
There are three phases in our empirical test: the creation of the test, the running of the test, and the evaluation of the test results. Each of these phases tries to be as easy and automated as possible by providing a quick empirical evaluation of a notification system thus addressing the problems proposed in the introduction. The systems we tested were based on Expedia.com. Students developed notification systems to monitor flight information, and used our empirical test system to evaluate how well they performed.

2.3.1 Creating the Test Setup

The goal of the test creation phase is to create the script file that drives the primary task in the dual-task environment. This phase guides the designer through the creation of a script file by asking a series of questions. Since we cannot feasibly "automatically" extract all the information needed to run a test from the prototype or design documents, we provide helpful comments throughout this process thus reducing the cost of learning. Designers are first given an overview of the automated empirical evaluator system. They are informed about what this stage does, what information it will collect, and what will become of the information that will be collected. If they do not know what their results will be like before being forced to do the setup, they will likely forego the whole system.

Since the setup phase is the first thing the designer sees, it begins with an overview of the system so the designer can judge how much work it is going to take and what benefits she will receive from it. The designer is then guided to select the environment (i.e. the primary task) for which to test the end users. For example, with the Expedia notification system, our participant designers chose a game where the end user moves a bar left and right to catch falling blocks. Other examples would be typing tasks or video games where the level of concentration on the primary task is variable. We termed this to be the 'cost of interruption'.

After entering the cost of interruption, the automated system asks the designer to set up practice, benchmark, and activity rounds. Practice rounds are important because they acclimate the end user to the primary task. Specifically, the user is working with the primary task but performance is not measured. Benchmark rounds are also important because performance is logged after acclimation occurs. This allows the automated system to have a performance measure of the end user that can be measured against after the secondary task is added. After entering the information



Using animation to get updates on status

Rating

☆★★★

Description

The NS is being used to support the planning-scheduling primary task by providing periodic informational updates to user. This information is located in a small space and updates are provided every so often as the particular topic is cycled through with other topics.

Upsides

+ Animation is used to display many different pieces of data in a limited area
+ Use of animation for updating is not intrusive and has low interruption.
+ Animation is predictable, so users have a rough idea of when a certain topic is coming up.
+ Users can control animation in desktop situations

Downsides

- When WH is on a large, common display, the user cannot control the animation and has to wait for the information he/she is interested to cycle around.
- There is no difference in how the information is presented, so there is no reflection of priority of information.
- Because this is primarily a desktop application, font may not be optimal for a large screen

Scenarios

Trudy works in a lab that is located in the center of her building. She usually works in this location, but some of her work involves doing to another building. During the winter's harsh weather conditions, Trudy would like to be informed of current weather conditions outside. The WH system can inform her of such information. Trudy can now continue to do her work. When she must leave to go to another building, she can briefly glance at the WH display and quickly interpret the weather information. Now she can prepare herself for the weather outside.

Figure 2. Example of a claim as it would appear in the library. (For reference: NS= Notification System, WH = Weather Highlighter)

needed for both of those rounds the designer enters information about and the actual version of the prototyped notification system for the activity rounds (i.e., how many rounds to have, when notifications will occur, and how long the notifications will last for). For example, with our Expedia participants the designer might enter that a new flight occurs with her particular prototype at forty seconds and a message is displayed for eight seconds. At all of these phases the designers are offered helpful information on why both of these phases are important and how they are going to affect the end users.

While entering in information about their prototype, designers also had claims they would enter (See Figure 2) that were populated from the LINK-UP Library. If they had multiple artifacts (ex: popups), they would have multiple notifications at different times associated with the same claim. This way, designers can setup the test platform to automatically log performance data captured during a signal as relevant to its associated claims.

The last phase of setting up the test allows the user to personalize the test to their particular prototype. We support this idea in two ways: dialog boxes and user defined questions. Through the use of dialog boxes the designer is allowed to enter in explanatory information for how their system is going to behave. The dialog boxes are a simple way for the designer to retain control over the automated test platform should she wish to do something that it does not support. For user defined questions we support the idea that the designer will want to generate qualitative data. For compatibility with our critical parameter measurements, there are two types of questions: comprehension and projection. A comprehension question is one that asks about a specific event or object, such as "What was the cheapest flight overall?" A projection question deals with trends and predicting new events, such as "Did flight prices increase or decrease overall?" or, "Would you expect flights next week to be cheaper or more expensive than this week?"

After inputting the questions, the user is done with the setup, and most of the work needed for the test in general. The work is heaviest early on, but no more so than any traditional manually run experiment. By guiding the designer through our setup, we hope to not only make it easier for expert designers, but possible for novice designers to run an empirical study without the expert knowledge.

2.3.2 Running the Test

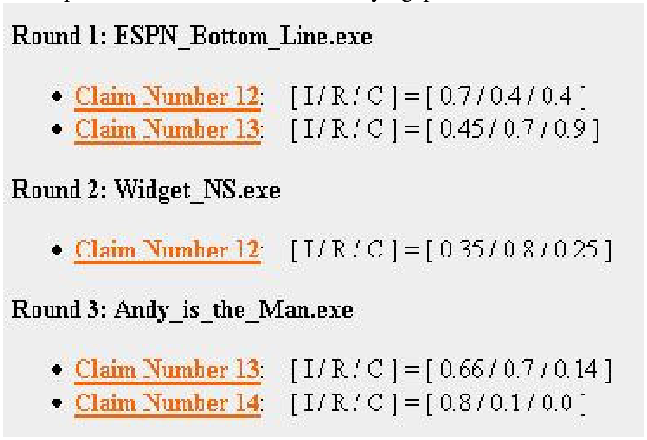
The testing environment is completely automated. Once the test begins, users can complete the test without any guidance or dealings with the designer. Instructions are included in the test from the setup phase. If desired, designers can have instructions be to "listen for verbal instructions" (as specified in section 2.3.1). All the data collection is handled by our back end. All the user interaction with the system during the test itself is logged, and questions are asked and answers recorded at the end. This eliminates the need for tedious pen and paper surveys, which requires one to collect them, enter all the data, and then run calculations on them.

2.3.3 Evaluating Test Results

Once the data has been uploaded, the LINK-UP user can view the results of each test she has run. Each round is subdivided into the claims that were tested for that round. The average I, R, and C values obtained for each claim are displayed next to the claim name (Figure 3). For further information regarding each claim, the user can view a claim detail page, which contains information such as a claim description, upsides, downsides, scenarios, expected I, R, and C values, and I, R, and C values obtained through empirical testing. These parameters are displayed in their raw format, a way that speaks naturally to the user (Figure 3). By appending the empirical results to the claims in the library, other designers can benefit from someone else's testing results. By sharing this knowledge, if a designer is not satisfied with her system's performance, she can hunt through the claims library to find a claim that suits her more.

2.3.4 Link Back to Reuse and LINK-UP

One of the primary goals of this system, and indeed of LINK-UP overall, is to support reuse. The automatic test platform itself is a reusable component of testing. Creating a testing environment for an interface is very time consuming. Problems with the test itself will prohibit the tester from identifying problems with her own interface should some error occur during data collection. A reusable test platform would not only save time, money, and potential data, but create a standard for which to measure interfaces as in [1]. Working within the scope of notification systems and using IRC parameters, we were able to identify many reusable components. First, to even be a notification system, there must be a primary task. By allowing substitutable primary tasks within our testing platform, they can be reused from test to test. Furthermore, every test needs data collection and visualization, both of which are handled by this module.


Round 1: ESPN_Bottom_Line.exe

- Claim Number 12: [I / R / C] = [0.7 / 0.4 / 0.4]
- Claim Number 13: [I / R / C] = [0.45 / 0.7 / 0.9]

Round 2: Widget_NS.exe

- Claim Number 12: [I / R / C] = [0.35 / 0.8 / 0.25]

Round 3: Andy_is_the_Man.exe

- Claim Number 13: [I / R / C] = [0.66 / 0.7 / 0.14]
- Claim Number 14: [I / R / C] = [0.8 / 0.1 / 0.0]

Figure 3. Viewing results from the test.

The second part of reuse happens when designers view their results using the LINK-UP system. They are provided the option of searching for new artifacts that solve problems encountered during testing. Suppose a designer's artifact turned out to be too interruptive. She could then search the database for an artifact with an interruption slightly lower than her own artifact had, but with other similar claims. This is truly the heart of LINK-UP's reuse: the claim database. By providing designers access to each other's design knowledge, they save resources and build better systems. The value of the database depends on the value and variety of claims that it stocks, which is why our automatic testing platform is so important to LINK-UP. If we can provide an attractive, easy to use system that provides empirical testing results, then designers will be more willing to use it and thus provide their own design knowledge for others to reuse. One of the principal barriers of reuse is that you must first have something to reuse, and that takes effort. By automatically appending to the database results that are in a standard format that makes them reusable (IRC), we can overcome that barrier and stock up on design knowledge.

3. TESTING

The empirical testing device created was placed through two phases of testing: an end-to-end form of testing and development, and consistency testing. Through the use of this two-phase testing procedure, an in-depth and controlled empirical testing device emerged.

3.1 Phase 1, End–to–End Testing

The first phase of testing used seven participants who created notification systems based on flight prices posted on Expedia.com. The designers, who were expert level computer users, built their systems with a claim-centric approach using IRC parameters. The designers setup their experiments using the setup tool, and then tested their systems using the empirical testing device with three to five different participants. This allowed the seven designers to use their own mental model of how the developed systems should respond based on the critical parameters the system was founded upon. Before the designers viewed their results using the empirical testing device, the designers were questioned about where and how to redesign the said systems. The designers then viewed the results of their data using the empirical testing device and speculated upon where to redesign. Through analyzing the pre and post data on how to redesign, improvement of redesign strategies could be measured. Also, to measure ease of use and satisfaction, a questionnaire was given to evaluate what the designers thought of the system.

This phase of testing probed the designers' receptiveness to the software tool, as well as seeing if the designer's view of their system matched the view of their system provided by the empirical testing tool. It is important for satisfaction purposes that the expert designers feel the tool provides redesign suggestions either as good as or better than their own. Also, as experts, their opinions provide a reasonable estimation of if the tool results actually are as good as or better than their own.

3.2 Phase 2, Consistency Testing

From phase one, a notification system experiment was selected and the data on how to redesign was used. Six participants, each with limited computer knowledge, evaluated empirical data on how the notification system performs from the testing phase. The participants then answered questions about where and how to redesign the system based on the experimentation data.

This phase of testing investigated the results from the system independent from the tool itself to see whether or not consistent results could be obtained across multiple participants. Besides checking for consistency with the software tool itself, the point where design knowledge is transferred from the tool to the human to be implemented in the next iteration is susceptible to variation in interpretation, and thus of high interest.

3.3 Hypothesis

For the first phase of testing, our hypotheses were:

- Designers would have a moderate level of satisfaction. Specifically, we looked for below an average of 3 on the surveys. A 7 point scale was used, where 1 was strong satisfaction and 7 was strong dissatisfaction.
- Designers would be able to come up with useful, valid claims that they would feel confident with implementing.

The hypotheses for the second phase of testing were:

- Designers would be consistent in their redesign concerns.

- Designers would be confident that their redesign issues would improve the system by matching the design IRCs more closely.

4. Results

Designers using the system in the first phase of testing expressed satisfaction with it overall, but did encounter some problems understanding parts of the setup (Figure 4). Designers of the system in the second phase were able to understand the system and make use of the results from the experiment, but only to an extent.

4.1 Phase 1

For the redesign survey, where designers were asked what and how they would redesign, we found out the following:

- 4 found new things to redesign that they would not have found without our system
- 3 confirmed earlier ideas they had about what to redesign or leave the same
- 6 said they either found something new or confirmed their old ideas about redesign.

Designers using the system agreed that the system was easy to use, interesting, stimulating, and satisfying (Figure 4). They also agreed that the system presented important findings. However, they did not feel that it would be important to redesign if the test results did not match their intended purpose (Figure 4).

The system simplifies the task of evaluating notification systems compared to other methods.	2.29
I was able to understand and provide the necessary information to the script generator for the section about creating signals.	4.43
I was able to understand and provide the necessary information to the script generator for the section about creating dialog boxes.	2.29
If the test result IRC does not match the IRC I designed for, I think it would be very important to redesign my interface or rethink my requirement assumptions.	4.17
Having IRC values that characterize user performance does not give me any useful information, if I were to continue designing this interface or others like it.	5.14
Testing my prototype with users provided important findings, either showing the interface was already well-designed or helping to identify weaknesses.	2.14
I found the system satisfying, interesting, stimulating, and easy.	2.82

Figure 4. Scores out of a 7-point Likert Scale: 1 meaning "Agree Completely"

4.2 Phase 2

Novice designers were able to understand the domain for the experiment, as they agreed that they understood notification systems in general (Figure 5). They also understood our method of IRC. However, they did not agree that IRC was effective enough to evaluate interfaces or redesign by itself, and did not feel confident that there redesigns would match the original intended IRC values given by the designer (Figure 5).

For the second phase, the redesign questionnaire revealed that out of a total of 18 responses (6 users across 3 claims), 6 of them cited IRC values specifically as justification for a redesign issue. These answers to the following question are below:

"Would you redesign the system based on the claim? If so, why and how?"

- "No, because the interruption is lower than originally anticipated and the main purpose of this method is to have low interruption."
- "No, because the anticipated downsides were high interruption, but empirical study showed that this method was not much more interruptive than the other methods."
- "I think reaction should be higher, a chime for a new flight would draw the user's attention. It would also reduce the chance of the user ignoring the program."
- "While the interruption and reaction components are lower than expected, I feel this to be fairly unimportant, though I believe comprehension should be higher."
- "I don't think the original values of 1/1/0.5 were reasonable. The system purchases tickets automatically, so no reaction or comprehension is necessary. Also, interruption isn't important because the user can respond at any time."
- "If you want more interruption, perhaps it would be useful to associate the popup with a noise, flashing, etc."

I understood what notification systems were.	2
I understood what IRC stood for.	2.44
The combination of I,R, and C is enough to evaluate the effectiveness of an artifact.	4.67
The combination of I,R, and C is enough to effectively redesign.	4.2
I was confident that my redesigned artifact would more closely match the designer's IRC.	4

Figure 5. Scores out of 7-point Likert Scale

5. DISCUSSION

Our hypothesis for phase one of testing was met, as users were satisfied with the system. With an average of 2.82 (more than slight agreement) the users indicated the system was satisfying, interesting, stimulating, and easy. Key to design reuse is the ability of a system to entice usage to overcome any inherent reluctance. However, the second hypothesis for the first phase one was not fully met: designers did find useful, valid claims, but did not feel comfortable implementing them over their own ideas. Designers said that they were neutral (4.17) about redesigning "if the test result IRC does not match the IRC [they] designed for." Convincing expert users to trust a machine over their own judgment is a difficult task. It is undetermined if the tool failed because the tool's suggestions for redesign were truly inferior to the expert designers' own ideas, or because the tool failed to instill confidence and trust in the user.

The hypothesis for phase two regarding consistent redesign issues was not met. Users gave a wide array of answers about where to redesign. This is not too surprising, as usability studies often capture different problems with each test, and different designers interpret the problems in unique ways as well. A more robust test may have included an expert review of the redesign suggestions to evaluate them for "correctness" as well as consistency. Had this hypothesis been met, the system would have conquered a large problem in usability testing: inconsistency. The second phase two hypothesis was not met, as users did not agree that IRC was enough to evaluate the interfaces and redesign them. Since they felt IRC was insufficient, it is somewhat moot that they did not think their redesign issues would match the design IRC more closely than the original interface.

Users of the system understood the evaluation topics and terms presented to them, and we are confident they understood the tasks set before them. Why then, did they not take the system results more seriously and trust them? The results were not perceived as accurate and thus not trusted, either due to a misconception or a true flaw in the system. We feel we have achieved a measure of user acceptance which will enable us to return to the drawing board to make a system which is already easy to use into a more robust one.

6. FUTURE WORK

Many of the usability problems that arose during testing were due to the designers' unfamiliarity with key terms that were used in the script creation phase. Exit surveys helped us realize that, contrary to what we had originally thought, the designers did not have a clear picture of what the empirical tests would entail. This relates to the idea that even with automation empirical design is a hard process to learn. Because of this, the first portion of future work that should be addressed is an end-to-end animated tutorial. This tutorial should walk a designer through the entire empirical test process – from creation to completion. The tutorial would cover all key terms and jargon used in the empirical test module. Also, it would show the designer how data entered in the script creation phase affects the empirical test. This testing should be coordinated by one designer over the course of semester or four months, with four or more students used as test subjects.

7. ACKNOWLEDGMENTS

Our thanks go to all those who worked on LINK-UP prior to, and concurrent with, our work: special thanks to Anderson Ray Tarpley III for his hard work on this project. Many undergraduate seminars and graduate classes have been hard at work on LINK-UP in the previous years, allowing us this chance to implement their design.

8. REFERENCES

[1] Booker, J. E., Chewar, C. M., & McCrickard, D. S. Usability Testing of Notification Interfaces: Are We Focused on the Best Metrics? *41st Annual ACM Southeast Conference.* (April 2004), pp. 128-133

[2] Carroll, J. M. Rosson, M. B. 2002. *Usability Engineering: Scenario-Based Development of Human-Computer Interaction.* San Francisco, CA: Morgan Kaufman.

[3] Carroll, J. M., Singley, M., Rosson, M. B. Integrating theory development with design evaluation. *Behavior & Information Technology*. 1992. 11(5). pp. 247-255

[4] Chewar, C. M., Bachetti, E., McCrickard D. S., and Booker, J. *Automating a Design Reuse Facility with Critical Parameters: Lessons Learned in Developing the LINK-UP System*. In *Proceedings of the 2004 International Conference on Computer-Aided Design of User Interface (CADUI 2004),* Island of Madeira, Portugal, January 2004, pp. 236-247.

[5] Fabian, A., Felton, D., Grant, M., Montabert, C., Pious, K., Rashidi, N., Tarpley, A. R., Taylor, N., Chewar, C. M., McCrickard, D. S. Designing the Claims Reuse Library: Validating Classification Methods for Notification System. *41th Annual ACM Southeast Conferenc*e. (April 2004). pp. 357-362

[6] Lee, J. C., Lin, S., Chewar, C. M., McCrickard, Fabian, A. Jackson, A. From Chaos to Cooperation: Teaching Analytic Evaluation with LINK-UP. *In Proceedings of the World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education (E-Learn '04)*, November 2004

[7] McCrickard, D. S., Catrambone, R., Chewar, C. M., Stasko, J. T. Establishing Tradeoffs that Leverage Attention for Utility: Empirically Evaluating Information Display in Notification Systems. *International Journal of Human-Computer Studies* 58(5), May 2003, pp. 547-582

[8] Newman, W. M. Better or Just Different? On the Benefits of Designing Interactive Systems in terms of Critical Parameters. *In Proc. of DIS* 97, ACM, 1997, pp 239-245.

[9] Norman, D. A. Cognitive engineering. In Donald A. Norman and Stephan W. Draper (eds), *User centered system design: New perspectives on human computer interaction.* Lawrence Erlbaum Associates, 1986, 31-62.

[10] Payne, C., Allgood, C. F., Chewar, C. M., Holbrook, C., McCrickard, D. S. Generalizing Interface Design Knowledge: Lessons Learned from Developing a Claims Library. *IEEE International Conference on Information Reuse and Integration*. (October 2003), pp. 362-369

[11] Rittle, H. W. J. On the Planning Crisis: Systems Analysis of the 'First and Second Generations'. Reprint #107 from *Bedrifts Okonomen*, no. 8 (October 1972), Berkeley, Institute of Urban and Regional Development, University of California.

[12] Whittaker, S., Terveen, L., and Nardi, B. (Let's stop pushing the envelope and start addressing it: a reference task agenda for HCI. *Human Computer Interaction* (2003), 15, 75-106