

Task Model Driven User Testing for Web Applications

Pasi Lehtimäki

September 29, 2006

Abstract

Contents

1	Introduction	5
2	User Testing for Web Applications	5
2.1	Problems in User Testing	5
2.2	Methodology Analysis	6
2.2.1	Analysis of Think Aloud Protocol	6
2.2.2	Analysis of Remote User Testing	7
2.3	Measuring Usability	7
2.3.1	Qualitative Usability Metrics	7
2.3.2	Quantitative Usability Metrics	8
2.4	Communication Gap in User Test Results	8
3	Task Modeling	8
3.1	What Is Task Modeling	8
3.2	Task Model Notation in This Study	9
3.3	Creating User Interfaces from Task Models	9
3.4	Task Models at Runtime	9
4	Improving User Testing with Task Models	10
4.1	Using Task Models in Preparation Phase	10
4.2	Using Task Models in Execution Phase	10
4.2.1	Tracking Scenario	10
4.2.2	Tools for Test Observer	11
4.3	Using Task Models After Tests	11
4.3.1	Analyzing the Test Results	11
4.3.2	Producing Test Reports	12
5	Introduction to SysOpen Digia Realizer	12
5.1	Design philosophy	12
5.2	Realizer Tool	12
5.2.1	Task Models	12
5.2.2	Views	12
5.2.3	Panes	12
5.3	Runtime Environment	12
6	User Testing Framework	13
6.1	Objectives	13
6.2	Test Scenario Composer	13
6.3	Test Execution Framework	13
6.4	Reporting Framework	13

7 Case Study	13
7.1 Introduction to the System Used in Case Study	13
7.2 User Test Scenarios	13
7.3 Test Users	13
7.4 Test Flow	13
7.4.1 Before Test	14
7.4.2 During Test	14
7.4.3 After Test	14
7.5 Communicating the Test Results to Stakeholders	14
7.6 Redesign of the Tested System	14
7.7 Analysis	14
8 Conclusions	14
9 Terminology	15

1 Introduction

Task modeling is emerging technique for application definition and implementation. Task modeling has great potential to solve problems in multi channel application development. It provides a way to formally model interactions between a user and an application. This technique is well suited for web application development because of the simple nature of web application control flow.

This study studies possibilities of task model driven systems for conducting user testing. The formality of task model based application development provides a way to formally define and track user test scenarios. The goal of this study is to introduce new tools for task model driven user testing and find out what kinds of usability metrics can be measured automatically using task model driven environment.

Section 2 contains an introduction and analysis of current methodology of user testing. In section 3 we provide an introduction to task modeling. Section 4 contains analysis of using task model driven system in different phases of user testing.

2 User Testing for Web Applications

Usability testing is a mandatory element in most software projects. Without adequate investment in usability, finished software will probably contain numerous usability defects or might even be found completely unusable by the end users. Usability defects can cause financial losses in form of lost customers or loss of effective work time by employees. Studies show that 80% of maintenance costs are related to problems users have with the product instead of technical defects. [2]

Focus in this study is on multi channel web application user testing. This study does not cover other forms of usability evaluations like heuristic evaluations and cognitive walkthroughs. Application domain under consideration is web applications with focus on mobile applications still not ruling out the traditional web applications aimed at desktop devices.

This section provides background about problems in user testing and evaluates two widely used methods for conducting the testing: think aloud protocol and remote user testing. Issues with measuring usability are covered in section 2.3. Communication problems between usability professionals and other groups of stakeholders are handled in the last section.

2.1 Problems in User Testing

User testing contains a lot of problems. As devices are getting smaller and mobile, the testing gets more complicated than it used to be with traditional stationary terminals. Conducting user tests in busy urban environment is quite different from conducting them in calm surroundings of a usability testing laboratory.[15] Simply seeing the screen of a small device can be a challenging task to a user test observer and things get even more difficult when the user navigates his or her way through busy metro stations or in other similar situations.

Currently there is a lot of debate going on about whether user testing should be conducted in field or not. Some researches say that same amount of usability issues can be found in laboratory studies than in field studies.[3] Other researchers on the other hand state that field testing is an essential way for finding critical usability issues when using mobile devices.[8]

The main problem in field studies seems to be that field testing is twice as time consuming as laboratory testing. Field testing could become more feasible if problems of mobility could be solved and expenses lowered to same or near the same level as the cost of laboratory tests.

Another problem is the need for a suitable prototype to be used in the user testing. Building a functional prototype may require a large amount of time and is usually discarded after testing. Low-fidelity prototypes on the other hand might not cope as well in the user testing or might not provide similar use experience as a finished product would.

The value of user test results highly depends on the quality of usability tasks used during testing. Selected tasks might be too simple or unrealistic.[21] A usability study might conclude that the product has no noteworthy usability defects when in fact the test scenarios were just too simple and did not correspond to real world use cases.

2.2 Methodology Analysis

In this section we analyze two widely used methods for conducting user testing. The methods have a different starting point but both have the same goal: to uncover as many as possible usability problems by letting test subjects use the product.

2.2.1 Analysis of Think Aloud Protocol

When people talk about user testing they most commonly refer to a method called think aloud protocol. Bottom line behind think aloud testing is that a test subject verbalizes his or her thoughts and allows the test observing personnel to understand the choices made during testing.

At the beginning of a test session a test observer presents a series of scenarios to the test subject. These scenarios are prepared by usability professionals to simulate common use cases of the product. The test subject then tries to carry out scenarios one by one.

In a successful test the test subject lets the usability professionals know what an ordinary user thinks when interacting with the product. This information can then be used to locate usability defects in the product and to propose a possible solution to fix them.

Conducting user testing using think aloud protocol is expensive and time consuming because most of the data recorded during testing is informal and has to be analyzed manually. After the analysis the usability professionals have to prepare reports and suggest redesigns as needed. This whole process is so lengthy that only few projects can afford many rounds of user testing or sufficient amount of test subjects.

Some researchers point out that usability professionals do some times design scenarios rather to confirm problems they already know instead of keeping eye on the product as a whole[21].

2.2.2 Analysis of Remote User Testing

Remote user testing means user testing where test subjects are not in a usability laboratory but instead perform the testing using devices and PCs in their common surroundings ie. at home or at work.

Remote testing methods can be roughly divided into two groups. One group depends on web meeting and remote screen capturing software to get insight on users' actions during testing. The other group relies on log analysis and lets the users to perform given tasks unsupervised.

The first group basically uses think aloud protocol remotely. The fact that test subjects do not have to be physically in the test laboratory lowers costs of the user testing and makes recruiting test subjects easier. This also allows user testing to be conducted using a wider audience around he world.

The second group has a different approach. They rely on automatically gathered usage log analysis and use large enough amount of test subjects to gather quantitative data about users' behavior.

Remote user testing becomes a lot more complicated when testing involves mobile devices. It is not feasible to run remote meeting software concurrently with actual tested software in mobile devices. Additional software could interfere with the real user experience and alter the actual outcome of the test.

To be able to conduct remote testing the testing group needs a functional prototype of the product. Paper prototyping or non-functional prototypes can't be used.

Some researchers indicate that it is possible to find nearly identical usability problems with remote testing methods than with traditional usability laboratory methods.[31]

2.3 Measuring Usability

Measuring usability in user testing can prove to be a difficult task. Qualitative data measurements often contain subjective opinions of usability professionals and because of the high expenses of user testing the quantitative data can not be measured effectively when the test is conducted using too few test subjects.

2.3.1 Qualitative Usability Metrics

Qualitative data is usually gathered manually by test observers. Carolyn Snyder[28](242-244) instructs all test observers to make notes during the tests. She also instructs everyone to divide notes into three categories straight away to avoid contentious debate during analysis phase. First category is *observations* which are used to describe facts. For example what user did or said. Second category, *inferences*, means

notes which are subjective inferences made by test observers during testing. The last category is *opinions*.

2.3.2 Quantitative Usability Metrics

The slowness of user testing tends to restrict the amount of test subjects as small as possible to cut down project expenses. The small amount of test subjects causes a lack of quantitative data since it can not be effectively measured from such a small number of tests. Using current methods for user testing it is not recommended even to use stop watch to record task completion times since the statistical error would render accurate measurements useless.[19] Nielsen suggests that for measuring any quantitative metrics the test would require at least 20 test subjects [23].

2.4 Communication Gap in User Test Results

Arguably the most important part of any usability evaluation is the part where findings are communicated to different stakeholders. Obviously the investment done in usability testing is in vain if the identified problems are not taken care of and usability of the product is not improved.

A very important group is the developers. Ultimately, they are responsible for implementing the changes. Some researchers point out that developers appreciate redesign proposal over just pointing usability problems in the original design [13].

Other groups of stakeholders might need fairly different sets of information about the usability evaluation.[32] Sales people might need just an quick overview of software quality as project leader needs more detailed estimate of how long it would take to fix the found defects and which defects are worth fixing for. One critical piece of information which might get overlooked is pointing out which kinds of assumptions can be made from the evaluation results. Sometimes people tend to think that all usability studies provide scientific facts about the product but that is not the case.[7] It is important to acknowledge that even when user testing with 5 test subjects seems to uncover a large portion of usability problems[22] it still can not prove the other way around. 5 persons can successfully completing every scenario in a user test do not mean that the tested product is error free.

3 Task Modeling

This study does not concentrate on task modeling as a research area but it is vital to understand some basic concepts of task modeling to understand the solution outlined in this paper. This section contains basic information about task modeling, how user interfaces can be implemented using task models and task models are used in a runtime environment.

3.1 What Is Task Modeling

Task modeling is a relatively new emerging method for defining users' interactions formally.[10] Formal task models can be used to describe product wide functionality

or just small and simple operations. Task modeling technique fits very well into web application domain because of the simplicity of control flow between a user and a web application.

3.2 Task Model Notation in This Study

This study uses a hierarchical task model notation developed for SysOpen Digia Realizer platform. Task models are composed of components called tasks. Each task represents either a user's operation or an operation performed by the system.

Task models contain three main types of tasks. 1) *User tasks*, which describe input from user to the system, 2) *visualization tasks* which describe system's output and 3) *system tasks* which describe operation performed by the system. Any number of these tasks can be wrapped into containers called *abstract tasks*. Each task is connected to its siblings with an operator. Operators can vary from simple flow describing operators to data submitting or choice operators.

Figure 1 illustrates a very simple task model in which system first displays some information to user and then waits for user's input. Once the user has made a selection the system then updates information.

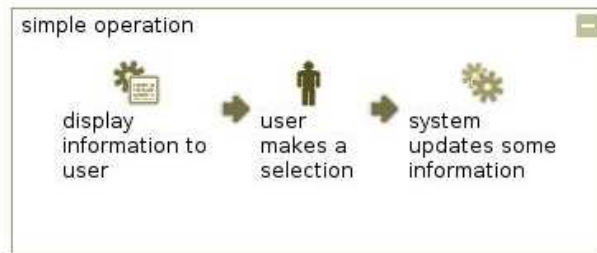


Figure 1: Simple task example

3.3 Creating User Interfaces from Task Models

Task models can be used to produce user interfaces for web applications. In abstract level it means that tasks describing user's input are displayed as links. Other tasks either cause back end operations or display data in user interface.

3.4 Task Models at Runtime

A runtime web application engine can be built around a task model engine. The engine will then be able to track task model execution while user uses the product. This functionality is the key for unearthing the real potential of task model driven user testing.

4 Improving User Testing with Task Models

The fact that task models can be the basis for runtime system execution provides new possibilities for improving user testing. The supporting idea behind task driven user testing is to use the same task models which function as the hearth of the product being tested to describe user test scenarios. This pairing unveils possibilities to formally track users' interactions within the product. This kind of task model driven system could ease user test analysis afterwards but also make it easier to design better test scenarios.

In this section we examine the ways task models could be used to aid in different phases of user testing.

4.1 Using Task Models in Preparation Phase

The way of designing user test scenarios has to be altered slightly to enable task model driven user testing. In addition to designing the scenarios it is mandatory to mark the starting point and the preferred goal task of each scenario into the task models used in the system.

After this the system can even give feedback about the designed scenarios automatically. It is possible to count test coverage and check for example how many possible error situations in the product are tested in those scenarios.

While designing user test scenarios usability professionals have to be careful not to depend solely on task models when considering use cases and users' goals. It might be tempting to do so since the task models do provide a very good overview of the product. The problem is that those task models are used to design and implement the product and if the user test scenarios are designed from the very same point of view the testing might not uncover those usability defects caused by failures to understand users' real goals in the design phase.

4.2 Using Task Models in Execution Phase

During the tests there are two clearly separate roles for the task models. Firstly the engine should do all the hard work for gathering required measurements and information of users' interactions to be used after testing and analyzed. The other role is providing the test observers tools to aid their work and allowing them to concentrate more on the user.

4.2.1 Tracking Scenario

Once a test subject starts to perform a test scenario the system starts logging. During the test the system records the path user uses and time used in each task. The system can also make note if some tasks are iterated more often than should be: for example if certain form proves to be too hard for users to fill the way it will pass all validation codes.

Once user reaches the selected target task the system records it but can not stop it there. The user might be confused whether or not he or she managed

to complete the given task. This kind of information is vital when analyzing the product's usability. There has to be way for a test subject to tell that he or she now thinks that the scenario task is completed or if the subject thinks he or she can not complete it at all.

4.2.2 Tools for Test Observer

Test observers have to be able to connect to the same system as the test subject performing the test. The system is able to provide the observer information about task model state and display a copy of test subject's current interface. A test observer can easily make notes during the test directly in context: all the observer has to do is to tell the system what the note is about and the system knows the state of task model being currently run and can mark the note directly into right context.

4.3 Using Task Models After Tests

After the testing task models can be really useful in many parts. Firstly task models ease the information extraction from the test since many events were logged automatically during tests and do not need to be analyzed manually. Other part where task models can prove to be valuable is test reporting. Usability problems can be marked directly into the task models which eases the communication between usability professionals and other stakeholders.

4.3.1 Analyzing the Test Results

Clear visualization of measured data can ease the analysis work which is done after the testing. The system can visualize users' paths during a user test scenario directly into the task models. Each choice can be paired with corresponding percentage of test subjects who selected the path. This allows test analyzers to locate immediately the most misleading parts of the tested product.

Also the average time needed to complete individual tasks can be visualized directly to the task models. For example if selecting a certain user task took a lot longer average time than the others it might not be presented clearly enough in user interface.

One very interesting area to be considered is multi channel products. The task model beneath different user interfaces for various devices is often the same. This makes it possible to compare users' performance using the same software with different kinds of devices. It is also possible to compare users' performance with the product in different kinds of surroundings.

List of measurable metrics in task model driven user testing

- Total time to complete a test scenario
- Time spent in each task

- Iterations in each task
- Percentage of each path used
- Percentage of successful completion of user test tasks

4.3.2 Producing Test Reports

Different groups of stakeholders need different kind of information from user tests.[27] It is vital to present the results for each group the way that the group needs it. Developers need a lot of information to be able to fix detected defects and improve product usability. In task model driven environment at least part of the defects can be pointed out directly in task models.

Sales people on the other hand do not need that detailed information about individual defects. Instead they need a summary and an overall report of what needs to be done. Project managers need to know how long it would take to fix critical problems and which problems have to be fixed before the product can be released.

5 Introduction to SysOpen Digia Realizer

5.1 Design philosophy

* uidl / tool / engine. * * solutions to multi channel / mobile web apps * * rapid prototyping *

5.2 Realizer Tool

5.2.1 Task Models

* task models * . . .

5.2.2 Views

Views describe the layout panes should take in each device.

* example view maybe? *

5.2.3 Panes

Panes describe the way user interface components should be presented.

* example pane maybe? *

5.3 Runtime Environment

Basis for understanding task model execution.

6 User Testing Framework

* design *

* don't think this needs really much technical details but instead give reader a proper understanding how it work in practice from the user's(not the test subject) point of view *

6.1 Objectives

* explain objectives here *

6.2 Test Scenario Composer

Way to create user test scenarios from task models

6.3 Test Execution Framework

Way to run tests and help test observers' work.

6.4 Reporting Framework

Ease reporting by providing quantitative data and pre filled report templates.

7 Case Study

A case where the framework was used

* if the implemented framework is operational early enough, use it in here. If not then describe how it will be used once ready based on theory and design *

7.1 Introduction to the System Used in Case Study

* explain the task model implementation * * introduce multi channel features used etc... *

7.2 User Test Scenarios

* list test scenarios *

7.3 Test Users

* who were used as test subjects *

7.4 Test Flow

* what was done before, during and after tests *

7.4.1 Before Test

* Explain things done before testing *

7.4.2 During Test

* Explain things done during testing *

7.4.3 After Test

* Explain post-testing procedures *

7.5 Communicating the Test Results to Stakeholders

* explain how test results were communicated to developers *

7.6 Redesign of the Tested System

* how test results were used to improve the system's usability *

7.7 Analysis

* Case study analysis *

8 Conclusions

How did the framework cope?

9 Terminology

Task Modeling

Task model	Task model is . . .
Task	Task is a single component in task model . . .
User task	User's input
Visualization task	System's output
System task	Operation performed by system
Abstract task	Collection of tasks

SysOpen Digia Realizer

Pane	Pane is . . .
View	View is . . .
Precondition	Device / role preconditions. . .

References

- [1] M. Donyaee A. Steffah and R. Kline. Usability and quality in use measurement and metrics: An integrative model. *Journal*, 2005.
- [2] Jonathan Benn Ahmed Seffah. Making usability a respectable quality attribute in the engineering lifecycle. *IEEE Canadian Review - Fall*, 2003.
- [3] Aki Kekäläinen Anu Kankainen Mihael Cankar Anne Kaikkonen, Anne Kaikkonen. Usability testing of mobile applications: A comparison between laboratory and field testing. *Journal of Usability Studies*, 1(1):4–16, November 2005.
- [4] Sandrine Balbo, Dirk Draheim, Christof Lutteroth, and Gerald Weber. Appropriateness of user interfaces to tasks. In *TAMODIA '05: Proceedings of the 4th international workshop on Task models and diagrams*, pages 111–118, New York, NY, USA, 2005. ACM Press.
- [5] Vince Bruno, Audrey Tam, and James Thom. Characteristics of web applications that affect usability: a review. In *OZCHI '05: Proceedings of the 19th conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction*, pages 1–4, Narrabundah, Australia, Australia, 2005. Computer-Human Interaction Special Interest Group (CHISIG) of Australia.
- [6] Ed H. Chi, Adam Rosien, Gesara Supattanasiri, Amanda Williams, Christiaan Royer, Celia Chow, Erica Robles, Brinda Dalal, Julie Chen, and Steve Cousins. The bloodhound project: automating discovery of web usability issues using the infoscentπ simulator. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 505–512, New York, NY, USA, 2003. ACM Press.
- [7] R. Stanley Dicks. Mis-usability: on the uses and misuses of usability testing. In *SIGDOC '02: Proceedings of the 20th annual international conference on Computer documentation*, pages 26–30, New York, NY, USA, 2002. ACM Press.
- [8] Henry Been-Lirn Duh, Gerald C. B. Tan, and Vivian Hsueh hua Chen. Usability evaluation for mobile device: a comparison of laboratory and field tests. In *MobileHCI '06: Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, pages 181–186, New York, NY, USA, 2006. ACM Press.
- [9] Maria R. Ebling and Bonnie E. John. On the contributions of different empirical data in usability testing. In *DIS '00: Proceedings of the conference on Designing interactive systems*, pages 289–296, New York, NY, USA, 2000. ACM Press.
- [10] Fabio Paternò Giulio Mori and Carmen Santoro. Ctte: Support for developing and analyzing task models for interactive system design. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 28(9), September 2002.

- [11] Erin E. Heximer, Uliyana Markova, Lisa Wu, and Justine Yoon. A multidisciplinary approach to improving the user experience: information development, test, and user experience design teams working together. In *SIGDOC '02: Proceedings of the 20th annual international conference on Computer documentation*, pages 72–78, New York, NY, USA, 2002. ACM Press.
- [12] Jason I. Hong and James A. Landay. Webquilt: a framework for capturing and visualizing the web experience. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 717–724, New York, NY, USA, 2001. ACM Press.
- [13] Kasper Hornbæk and Erik Frøst. Comparing usability problems and redesign proposals as input to practical systems development. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 391–400, New York, NY, USA, 2005. ACM Press.
- [14] SARAH WATERSON JAMES A. LANDAY JASON I. HONG, JEFFREY HEER. Webquilt: A proxy-based approach to remote web usability testing. *ACM Transactions on Information Systems*, 19(3):263–285, July 2001.
- [15] Melanie Kellar, Derek Reilly, Kirstie Hawkey, Malcolm Rodgers, Bonnie MacKay, David Dearman, Vicki Ha, W. Joseph MacInnes, Michael Nunes, Karen Parker, Tara Whalen, and Kori M. Inkpen. It's a jungle out there: practical considerations for evaluation in the city. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1533–1536, New York, NY, USA, 2005. ACM Press.
- [16] Jesper Kjeldskov, Mikael B. Skov, and Jan Stage. Instant data analysis: conducting usability evaluations in a day. In *NordiCHI '04: Proceedings of the third Nordic conference on Human-computer interaction*, pages 233–240, New York, NY, USA, 2004. ACM Press.
- [17] Tobias Klug and Jussi Kangasharju. Executable task models. In *TAMODIA '05: Proceedings of the 4th international workshop on Task models and diagrams*, pages 119–122, New York, NY, USA, 2005. ACM Press.
- [18] Matthias Krauss; and Dennis Krannich. ripcord: rapid interface prototyping for cordless devices. In *MobileHCI '06: Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, pages 187–190, New York, NY, USA, 2006. ACM Press.
- [19] Mike Kuniavsky. *Observing the User Experience: A Practitioner's Guide to User Research*. Morgan Kaufmann, 2003.
- [20] Judith A. Ramey Marcia A. Ruthford. The design response to usability test findings: A case study based on artifacts and interviews. *IEEE*, 2000.
- [21] Mie Nørgaard and Kasper Hornbæk. What do usability evaluators do in practice?: an explorative study of think-aloud testing. In *DIS '06: Proceedings of the 6th ACM conference on Designing Interactive systems*, pages 209–218, New York, NY, USA, 2006. ACM Press.

- [22] Jakob Nielsen. Why you only need to test with 5 users. <http://www.useit.com/alertbox/20000319.html> (referenced on 23.9.2006), March 2000.
- [23] Jakob Nielsen. Quantitative studies: How many users to test? http://www.useit.com/alertbox/quantitative_testing.html (referenced on 24.8.2006), June 2006.
- [24] Janni Nielsen, Torkil Clemmensen, and Carsten Yssing. Getting access to what goes on in people's heads?: reflections on the think-aloud technique. In *NordiCHI '02: Proceedings of the second Nordic conference on Human-computer interaction*, pages 101–110, New York, NY, USA, 2002. ACM Press.
- [25] Stephen Owen, Pearl Brereton, and David Budgen. Protocol analysis: a neglected practice. *Commun. ACM*, 49(2):117–122, 2006.
- [26] Judith Ramey, Ted Boren, Elisabeth Cuddihy, Joe Dumas, Zhiwei Guan, Maaïke J. van den Haak, and Menno D. T. De Jong. Does think aloud work?: how do we know? In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*, pages 45–48, New York, NY, USA, 2006. ACM Press.
- [27] Janice (Ginny) Redish, Randolph G. Bias, Robert Bailey, Rolf Molich, Joe Dumas, and Jared M. Spool. Usability in practice: formative usability evaluations - evolution and revolution. In *CHI '02: CHI '02 extended abstracts on Human factors in computing systems*, pages 885–890, New York, NY, USA, 2002. ACM Press.
- [28] Carolyn Snyder. *Paper Prototyping*. Morgan Kaufmann, 2003.
- [29] Kenia Sousa and Elizabeth Furtado. From usability tasks to usable user interfaces. In *TAMODIA '05: Proceedings of the 4th international workshop on Task models and diagrams*, pages 103–110, New York, NY, USA, 2005. ACM Press.
- [30] Jared Spool and Will Schroeder. Testing web sites: five users is nowhere near enough. In *CHI '01: CHI '01 extended abstracts on Human factors in computing systems*, pages 285–286, New York, NY, USA, 2001. ACM Press.
- [31] Katherine E. Thompson, Evelyn P. Rozanski, and Anne R. Haake. Here, there, anywhere: remote usability testing that works. In *CITC5 '04: Proceedings of the 5th conference on Information technology education*, pages 132–137, New York, NY, USA, 2004. ACM Press.
- [32] Whitney Quesenberry Title. Towards the design of effective formative test reports. *Journal of Usability Studies*, 2005.
- [33] Ryan West and Katherine Lehman. Automated summative usability studies: an empirical evaluation. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 631–639, New York, NY, USA, 2006. ACM Press.

- [34] Andreas Wolff, Peter Forbrig, Anke Dittmar, and Daniel Reichart. Linking gui elements to tasks: supporting an evolutionary design process. In *TAMODIA '05: Proceedings of the 4th international workshop on Task models and diagrams*, pages 27–34, New York, NY, USA, 2005. ACM Press.