

---

# Framework for Comparative Usability Testing of Distributed Applications

**Kari Kostiainen**

Nokia Research Center  
Helsinki, Finland  
kari.ti.kostiainen@nokia.com

**Ersin Uzun**

University of California, Irvine  
CA, USA  
euzun@uci.edu

**Abstract**

Comparative usability tests are useful when the optimal user interface needs to be selected from several alternatives. However, performing comparative usability tests is laborious, especially for distributed applications. In this extended abstract we present a usability test framework for one distributed application type: pairing methods. Using this framework developing new user interfaces for pairing methods and testing them for usability is easy and fast. Our framework also supports automated test sessions, event logging and error condition simulation.

**Keywords**

Usability testing tool, comparative usability testing, distributed applications, pairing methods.

**ACM Classification Keywords**

H5.2 User interfaces: Evaluation/methodology.

**Introduction**

When an optimal user interface needs to be selected from more than one alternative proposals, e.g. for standardization or product development, a comparative usability test is useful to find out which user interface is the best.

Usability tests are performed with either complete implementations or with user interface mockups

created just for the usability test at hand. Typically, existing user interface implementations do not support testing features, such as automatic event logging, but an external tool has to be used. Support for testing features can be added to each existing implementation and mockup, but this requires a lot of repetitive work.

Performing usability tests for distributed applications is even more laborious. If existing implementations are not available, implementing mere user interface mockups is not enough; at least a partial implementation of the networking part is required since the real user experience of a distributed application depends on the interplay of software in two or more devices.

In this extended abstract we present a usability test framework for one particular distributed application type: pairing methods. Using this framework a usability test organizer can easily develop new pairing method user interfaces for usability testing without having to do any network programming; the framework handles device discovery, synchronization and inter-communication of user interfaces on different devices. The framework also takes care of event logging, supports automated test sessions and enables easy error condition simulation. Existing user interface implementations can be plugged in to the framework with minimal changes as well.

### **Related work**

Several software tools have been developed to automate usability testing of websites and standalone applications. To mention a few, Automated Summative Evaluation (ASE) [1] is a website testing tool that has a separate control window which is used to present browsing tasks to the test person. ASE captures events

when the user performs these tasks and automatically collects user feedback for each task. Test Environment Automation (TEA) [2] is another and somewhat similar tool for testing websites. GUITESTER [3] is a Windows application testing tool that records and analyzes user events and visualizes the results.

### **Problems and requirements**

None of the existing tools address a) organizing *comparative usability tests* and b) the specific problems related to testing *distributed applications*. An ideal automated usability test solution for distributed applications should provide:

1. Easy implementation of new user interfaces for usability testing without any network programming
2. Possibility to use existing implementations with as little changes as possible
3. Easy simulation of errors, since it is important to know how users behave in error conditions, especially when testing security software
4. Automated test sessions so that several user interface alternatives can be tested conveniently
5. Automated event logging and data analysis

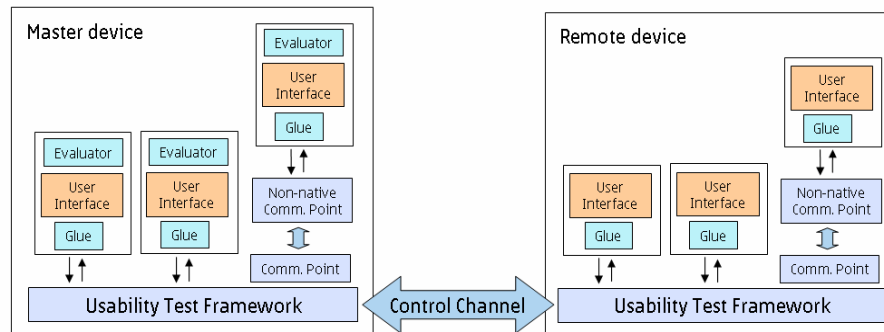
### **Device pairing**

Setting up a secure wireless connection between two devices is a common and challenging task for users. Recently, various different methods have been proposed both in research literature and standardization to tackle this *device pairing* problem. In all of the proposed methods the user needs to perform some user interaction, such as entering a short numeric value into both of the devices or comparing two short numeric values shown on the devices.

In some of the pairing methods, the user's inability to carry out the needed user interaction leads merely to unsuccessful pairing attempt, but in other methods the failed user interaction might cause a pairing with an unintended and possibly malicious device. Thus, an extensive comparative usability study is needed to find out which method is the easiest and which is the least error prone.

### Usability test framework

In this section we describe a *usability test framework*, an automated solution for usability testing of different pairing methods. The framework is installed into two test devices (see Figure 1). A *master device* orchestrates the test session using a *control channel* connection to a *remote device*.



**Figure 1.** Usability test framework architecture.

In the beginning of a test session, the test organizer determines the order in which different user interfaces are tested (random order is one alternative) and whether error conditions, such as man-in-the-middle attacks, should be simulated. Then, the master framework starts pairing user interfaces on both devices with a random input value. Each user interface

has its own way of using the input value, such as showing it to the user or asking the user to compare it with the value shown on the other device. The framework can simulate error conditions simply by starting user interfaces with different input values.

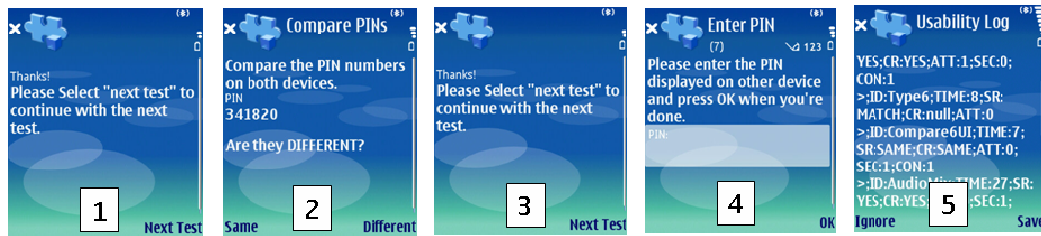
Once the user interaction with the tested user interface is finished, the test results from both devices are analyzed at the master device and a log entry is generated. After all user interfaces are tested, the test organizer may read the test results from the log file created on the master device.

To create a new user interface that is compatible with the framework, the developer needs to create a normal functional user interface and additionally implement two extra software modules: *glue* and *evaluator object*. The glue is a simple wrapper that ties the user interface implementation to the framework by implementing a few pre-defined functions. The evaluator object is used by the master framework to analyze the results. Each user interface has to have its own evaluator since the test results are specific to each tested user interface.

The framework also supports plugging in existing implementations. If the existing implementation is written in different programming language than the framework a *non-native communication point* needs to be implemented. This module converts function calls to local socket messages and vice versa.

### Implementation

We have implemented the framework using Java MIDP and tested it on Series 60 mobile phones. The control channel in our implementation is Bluetooth. The framework implementation is open sourced and available in <http://sconce.ics.uci.edu/CUF>.



**Figure 2.** Example usability test session.

An example test session is illustrated in Figure 2. The framework guides the test user through the test session and different pairing methods are tested one by one (1-4). After the whole test session has ended the test organizer may read the results from the log file created on the master device (5).

### Analysis

Compared to the traditional way of implementing user interface mockups our framework makes organizing usability tests for distributed applications considerably easier. The test organizer does not have to do any complicated network programming, since the framework takes care of all connectivity issues.

We have implemented several pairing method user interfaces for the framework and used the framework in real usability test of pairing methods [4]. We have also plugged in one existing pairing implementation [5] written in another language to the framework. Based on these experiences we can say that prototyping different pairing method user interfaces and testing them with real people is fast using the framework.

The ability to simulate error conditions provides an easy way to test how users react to unexpected conditions and how carefully they do read the instructions shown on the screen. The error condition

simulation is considerably easier than implementing a real source of errors, such as a functional man-in-the-middle attacker, for each test user interface separately.

### Future work

Two directions for continuing this work could be considered. 1) The presence of the test organizer and the whole test situation are likely to affect the test users' behavior, and thus the most reliable results are achieved if the tests are integrated to the everyday use of the device. Could the framework be extended to support this kind of long-lasting unattended user tests? 2) Currently the framework supports only pairing methods. How the framework could be generalizing for all kinds of distributed applications?

### Acknowledgements

We thank N. Asokan and Philip Ginzboorg for their valuable ideas and comments.

### References

- [1] Ryan West, Katherine Lehman. Automated Summative Usability Studies: An Empirical Evaluation. In Proc. of CHI 06, pages 631-639. April 2006.
- [2] Hartmut Obendorf, Harald Weinreich, Torsten Hass. Automatic Support for Web User Studies with SCONE and TEA. In Proc. of CHI 04, pages 1135-1138. April 2004.
- [3] Hidehiko Okada, Toshiyuki Asahi. Guitester: A log-based usability testing tool for graphical user interfaces. IEICE Transactions on Information and Systems. Vol. E82-D, no. 6, pages 1030-1041. 1999.
- [4] Ersin Uzun, Kristiina Karvonen, N. Asokan. Usability Analysis of Secure Pairing Methods. USEC 07: Usable Security. February 2007.
- [5] Nitesh Saxena, Jan-Erik Ekberg, Kari Kostianen, N. Asokan. Secure Device Pairing based on a Visual Channel. IEEE Symposium on Security and Privacy. May 2006.