

---

# 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

**N. Asokan**

Nokia Research Center  
Helsinki, Finland  
n.asokan@nokia.com

**Philip Ginzboorg**

Nokia Research Center  
Helsinki, Finland  
philip.ginzboorg@nokia.com

**Abstract**

Comparative usability tests are useful when the optimal method to perform a certain user task 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 pairing method mockups for usability testing 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. C2.4 Distributed Systems: Distributed applications/security.

**Introduction**

When the optimal method to perform a certain user task needs to be selected from more than one alternative proposals, e.g. for standardization or product development, two separate usability issues should be considered: 1) what is the best *user interaction model* for this task and 2) what kind of *user interface* presents this model in an optimal way to the

user. A comparative usability test is useful to find out both of these issues.

Traditionally, usability tests are done with either complete application implementations or with user interface mockups created just for the usability test at hand. Functionalities that support testing, such as automatic logging and measurement of timings, are typically not part of existing implementations but an external tool has to be used. Test functionalities can also be added to each existing implementation and mockup, but this requires a lot of repetitive work.

Doing 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 as well, because the real user experience of a distributed application depends on the interplay of software in two or more devices, and thus synchronization of user interfaces on different devices is needed.

In this extended abstract we present a usability test framework for pairing methods. Pairing method is a distributed application using which a person can set up a new and secure wireless connection between two devices. Using this framework a usability test organizer can easily develop new *pairing method mockups* for usability testing without having to do any network programming; the framework handles device discovery and synchronization of user interfaces on different devices. The framework also takes care of event logging, supports automated test sessions and enables easy error condition simulation. Moreover, existing pairing method implementations can be plugged into the framework with minimal changes.

## 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 for presenting browsing tasks to the test user. 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 usability test solution for distributed applications should provide:

1. easy implementation of new *distributed application mockups* for usability testing without any network programming,
2. possibility to use existing implementations in testing 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 alternatives can be tested conveniently, and
5. automated event logging and data analysis.

## Device pairing

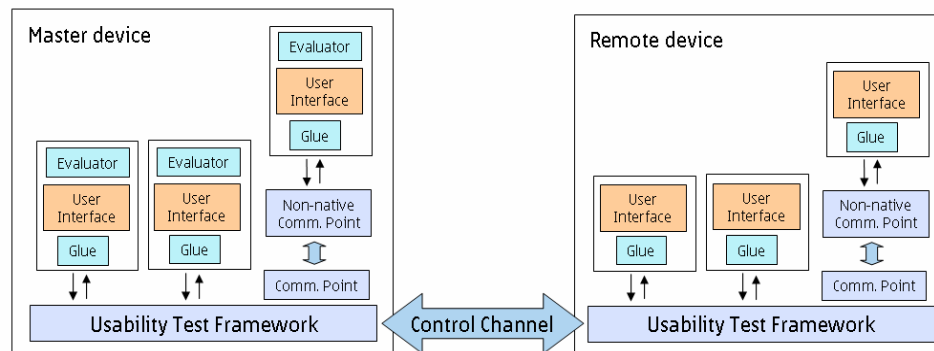
Setting up a secure wireless connection between two devices is a common and challenging task for users. Recently, various methods have been proposed both in

research literature and standardization to tackle this *device pairing* problem [4]. All of the proposed methods require 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 an 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 user interaction model is the easiest for users and what kind of user interface presents this model in the least error prone way.

### Usability test framework

In this section we describe a *usability test framework*, an automated solution for usability testing of 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 pairing methods are tested (e.g., random order is one alternative) and whether error conditions, such as those that would occur during a man-in-the middle attack, should be simulated. Then, the master framework starts pairing methods on both devices by sending command over the control channel. The pairing methods are started with an input value generated by the framework. Each pairing method 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 pairing methods with different input values.

Once the user interaction is finished on both devices, the master framework analyzes the results using an *evaluator* of that pairing method. The results of each pairing method are specific to the method at hand, and thus a separate evaluator must be provided with each pairing method mockup implementation (also a small *glue* module is needed to tie the user interface of the pairing method to the framework). Once the whole test session is completed, i.e. each pairing method has been tested, the master framework creates an entry to the local log. The test organizer may read the results from the master device after the test session.

The framework also supports plugging in existing implementations of pairing methods. 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>.

An example test session is illustrated in Figure 2. The framework guides the test user through the test session (1, 3) during which different pairing methods are tested one by one (2, 4). After the whole test session has been completed the test organizer may read the results from the master device (5).

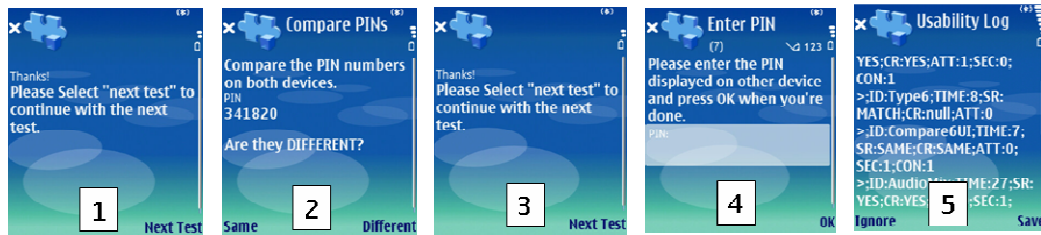


Figure 2. Example usability test session.

## Analysis

Our framework makes organizing usability tests for distributed applications considerably easier compared to the traditional way of implementing user interface mockups. 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 mockups for the framework and used it in real usability tests [5]. We have also plugged in one existing pairing method implementation [6] written in another language to the framework. In our experience prototyping different

pairing methods and testing them with real people using the framework is fast and easy.

The ability to simulate errors provides an easy way to test how users react to unexpected conditions and how carefully they 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 tested pairing method 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 behavior of test users, 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 can only be used to test different pairing methods. How the framework could be generalized for all kinds of distributed applications?

## References

- [1] R. West et al. Automated Summative Usability Studies: An Empirical Evaluation. In Proc. CHI 06, pp. 631-639. 2006.
- [2] H. Obendorf et al. Automatic Support for Web User Studies with SCONE and TEA. In Proc. CHI 04, pp. 1135-1138. 2004.
- [3] H. Okada et al. Guitester: A log-based usability testing tool for graphical user interfaces. IEICE Transactions on Information and Systems. Vol. E82-D, no. 6, pp. 1030-1041. 1999.
- [4] J. Suomalainen et al. Security Associations in Personal Networks: A Comparative Analysis. Nokia Research Center Technical Report. NRC-TR-2007-004. 2007.
- [5] E. Uzun et al. Usability Analysis of Secure Pairing Methods. USEC 07: Usable Security. 2007.
- [6] N. Saxena et al. Secure Device Pairing based on a Visual Channel. IEEE Symposium on Security and Privacy. 2006.