Tracking Human-Centric Controlled Experiments with Biscuit

Fernando Olivero[1], Michele Lanza[1], Marco D'Ambros[1], Romain Robbes[2]

1: REVEAL @ Faculty of Informatics, U. of Lugano, 2: PLEIAD @ DCC - University of Chile

Abstract

Software is created by humans, for humans. For this reason, software engineering is—above all—a human activity. Acknowledging this fact, many researchers perform controlled experiments with human subjects to evaluate the performance and usability of novel approaches and software engineering tools. However, the intrinsically non-deterministic nature of humans introduces a number of threats to the validity of such experiments. One of them concerns how to record information without influencing the behavior of the subjects involved. Another one relates to providing means to assure the correctness of the gathered data, for further pristine analyses and replication.

We present *Biscuit*, a tool that silently records relevant pieces of information regarding an experiment performed with human subjects. We present the main features and benefits of Biscuit by showcasing a controlled experiment of Gaucho, a next generation IDE. Based on our experience, we discuss the potential of Biscuit and outline future research in this direction.

1. Introduction

If software engineering is above all a human activity, then taking the human aspect out of the loop—when it comes to evaluating software engineering approaches—would be definitely wrong. Consequently, these past years have seen a steady increase of evaluations based on controlled experiments performed with human subjects; an example is the one of Cornelissen et al. [1]. Such experiments are prone to a large number of pitfalls, due to the one and only uncontrollable element of any controlled experiment involving human subjects: humans. Humans have individual talents, skills, behaviors, and quirks. In any experiment humans behave in a non-deterministic way, which introduces various threats to the validity of any experiment of this kind.

We recently performed an extensive controlled experiment [2] ourselves and, apart from the vast amount of energy and time such an endeavor entails—whose sensibility is not under discussion here—we noticed during our experiment a number of pitfalls that we want to alleviate with the work presented in this paper. The pitfalls in question regard an issue that might be seen as a corollary to Heisenberg's uncertainty principle: *How is the information that one wants to record actually being recorded, and does the fact that one records information in an intrusive way influence or modify the behavior of subjects? If so, how can this be minimized?*

In this paper, we present *Biscuit* [1], a toolset to:

1. specify tasks to be undertaken by the subjects of the experiment
2. precisely time the subjects and record their complete behavior as they perform the tasks,
3. store the answers the subjects give upon task completion,
4. provide data that is easy to process further, and send the data back to the experimenters.

We do not address other important issues pertinent to controlled experiment involving humans, e.g., the choice of participants, controlled variables, tasks, etc. These and others are all important questions which go beyond the scope of the present paper. We focus on a seemingly smaller set of issues, which however can and does contribute to a loss of precision or even a falsification of the recorded data.

2. The Crux of Human-centric Experiments

There are diverse possibilities to record information in a controlled experiment, where a usual scenario is that a set of subjects are divided into two groups, the control group and the experimental group. Subjects in the former group are given some baseline setting, while the subjects of the latter group are given the tool to be evaluated. The goal is then to assess whether the experimental group can perform a set of tasks better, faster, etc. as opposed to the control group.

A literature survey[2] ([1, 3]) reveals commonly adopted approaches to tackle the four issues we are focusing on in this paper, with which every experimenter is confronted:

[1] Available at `http://www.inf.usi.ch/phd/olivero/biscuit`

[2] We only cite a few examples, see [2] for a discussion of related work.

1) Give the subjects tasks to perform and/or questions to answer, and the possibility to provide answers/findings in some form. The subjects know what is expected from them; this can either come as a set of questions or a set of tasks. After performing the tasks the subjects then provide some form of feedback about what they have done, i.e., they need to answer questions. The experimenter's role in this situation is then, to obtain data about the correctness (i.e., was a task fulfilled or not/only partially) and the time taken by the subjects to perform a task. In the large majority of cases, experimenters opt for questionnaires, (often in the form of multiple choice questions or free-form text questions) to the subjects who fill them out during the experiment. Questionnaires can either be on paper or also in electronic form (an often adopted solution is to use survey websites).
2) Keep track of the time taken by each subject on each task. How can one reliably record the time it took for a subject to perform a task? One possibility is to have the subject write down the time (as part of the questionnaire); this introduces the risk of subjects writing down wrong information. Another possibility is for the experimenter to record the time information, which makes it hard, if not impossible, to perform an experiment with multiple subjects at the same time or a remote experiment—without removing measurement issues due to human fallibility.
3) Record what the subjects do while they perform tasks to try to find answers. To record what subjects do, the often adopted solutions are *Think-aloud protocols* and/or *filming* and *Screencasts* and/or *audio-recording*.

Think-aloud protocols involve experiment participants thinking aloud as they are performing a set of specified tasks. Users are asked to say whatever they are looking at, thinking, doing, and feeling, as they go about their task. This enables observers to see first-hand the process of task completion (rather than only its final product). Observers at such a test are asked to objectively take notes of everything that users say, without attempting to interpret their actions and words. Test sessions are often audio and video taped so that developers can go back and refer to what participants did, and how they reacted. The purpose of this method is to make explicit what is implicitly present in subjects who are able to perform a specific task.

A screencast is a digital recording of computer screen output, also known as a video screen capture, often containing audio narration. Experimenters can use them to record the full interaction of the subjects with the tools they used.
4) Process the previously recorded data to extract additional information, and allow the experimenters to gather the data easily. An additional problem is that the data recorded using such approaches needs to be post-processed since it comes as a digital movie/audio recording. The post-processing, e.g., transcribing what happened, can be lengthy and imprecise—especially when the number of subjects is high—due to the lack of formalism in natural language and human behavior. Raising the level of abstraction of the data would allow for easier and more automated processing. Related to this issue, gathering the data in itself can be a challenge—either because of manual processing of hand-written questionnaires, or because it comes from multiple sources (e.g. questionnaires, timing data, and other recorded data).

Summing up. Due to issues related to timing, data processing, and mostly inaccurate tracking of what happens during the experiment, the risk is that the data that is being collected during an experiment is distorted or even wrong. To mitigate such risks we devised *Biscuit*, a tool infrastructure for non-intrusive and precise tracking of data related to experiments with human subjects, presented next.

3. Biscuit: Precise Data Tracking

Biscuit supports performing controlled experiments, by recording relevant pieces of information regarding an experiment performed with human subjects. At the moment, it is geared towards experiments evaluating development tools.

To address the first issue stated in the previous section, Biscuit can set up an experiment made up of tasks which in turn, consist in a description and goals that need to be accomplished by the subjects. The format of answers to the goals range from multiple choice questions and free form text entries, to modifying the underlying system entities—such as adding/removing classes/methods until automated tests pass. Using *Biscuit*, the experimenter can automatically generate a user interface for any experiment; it presents the experiment to the subjects and guides them through the tasks until completion, storing the answers and durations of each task transparently. This addresses the first two issues; namely giving the subjects tasks to perform or questions to answer; and registering the answers and completion times.

To address the third issue—*i.e.,* recording—prior to running each task, *Biscuit* installs a spy that records *every user interaction*: from simple mouse move events, to more complex interactions such as performing changes to the code base, or providing answers to the experiment goals. The spy is built on top of a system monitoring tool called Spyware [4], enhanced with a complete instrumentation of the Event-Command pattern. The recording enables one to keep track of every user interaction in the form of recorded events that trigger commands, thus eliminating a great level of uncertainty from the correctness of the subjects answers, due to the *faithful replicability* of each experiment run. The events recorded by Biscuit correspond to actual user actions (source code navigation and modifications), in contrast to video recordings that require significant further interpretation, thus addressing the fourth issue: the event trace is open to automated analyses and can be replicated.

To address the fourth and final issue, upon completing an experiment the subject is asked to send (via email) an automatically created zipped file containing all the recorded data to the experimenters. This file contains every user interac-

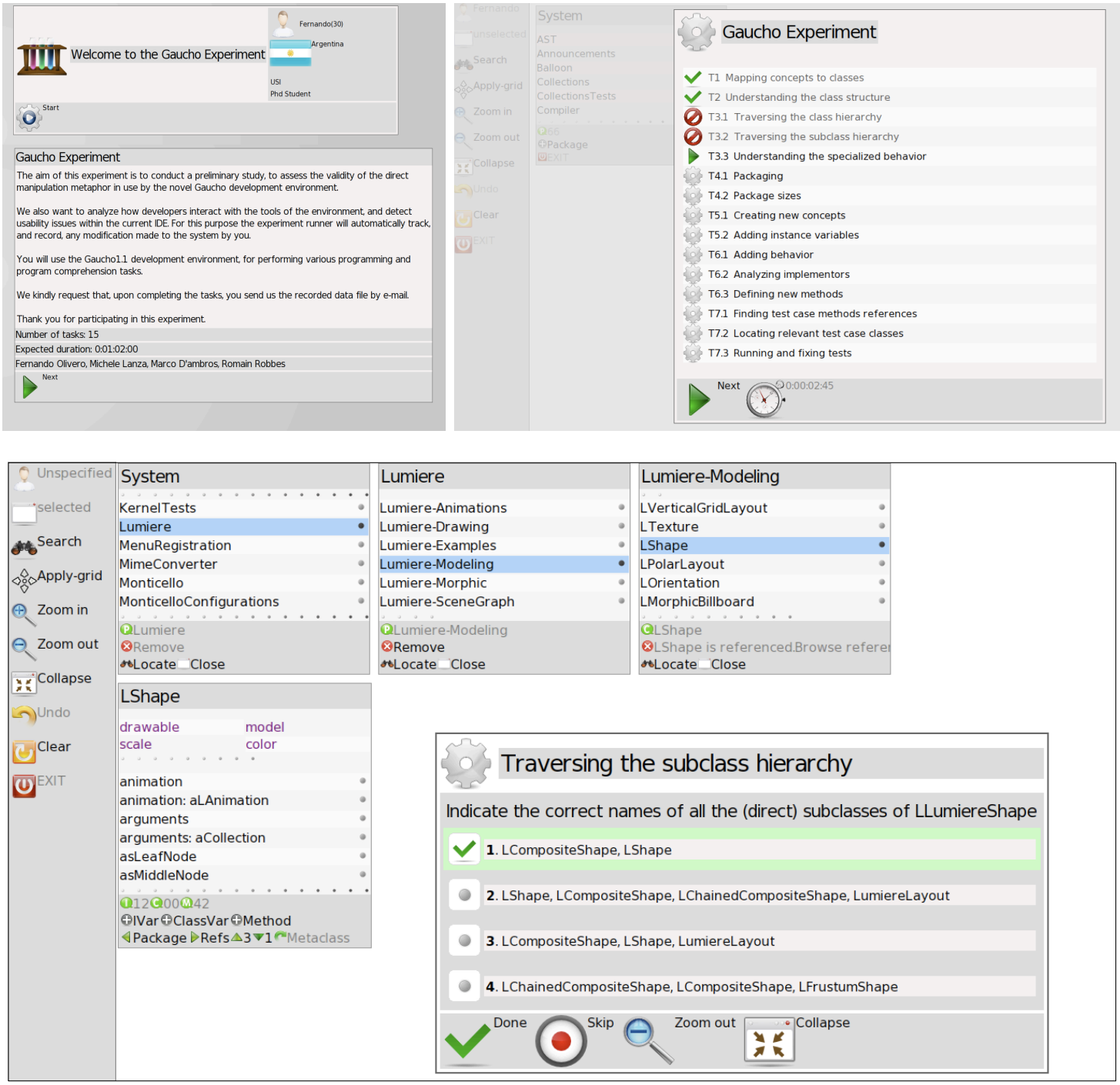

Figure 1. A Biscuit task list and a running task example, overlaid on top of the Gaucho IDE

tion recorded by the spy during the experiment run, together with the answers and solutions to each task, and additional meta-data such as the participant name or identifier, and the total completion time. The process is straightforward both from the subject's and the experimenter's point of view.

First test run. We conducted a preliminary evaluation of Biscuit, by setting up a controlled experiment to assess the validity of the metaphor in use by Gaucho—a development environment for software based on direct manipulation [5].

The goal of the experiment was to compare Gaucho with a standard Smalltalk IDE, the Pharo development environment (`http://pharo-project.org`) with respect to performing common development tasks, such as creating, navigating, refactoring, and understanding object-oriented code.

A secondary goal was to analyze how developers interact with the respective tools, to detect usability issues and provide insights into further improvements. We presented the experiment and our findings in [6], which was fully instrumented using Biscuit.

Figure 1 depicts the experiment description that is shown once the subject has completed the pre-test questionnaire and entered basic data, and an actual experimental run. The top part depicts the list of tasks to be done (two completed, two skipped), while the bottom part shows an actual task being performed. The screenshot illustrates how the Biscuit task runner records and informs users of the passage of time, presents the tasks, and collects the answers from the subjects. On the bottom corner, we see the task widget with the answer form; the rest of the screen is occupied by the tool being evaluated, in this case Gaucho. Figure 2 depicts the post-experiment questionnaire and the final widget presented to the subject.

Figure 3 depicts the contents of the files that constitute the output of an experiment run. Biscuit records all the data pertinent to the subjects activity, ranging from meta-data such as the level of expertise of the subjects, to low level user interface actions, such as mouse clicks and keystrokes. We automatically analyzed the results of the experiment by extracting the subjects' activity from these output files.

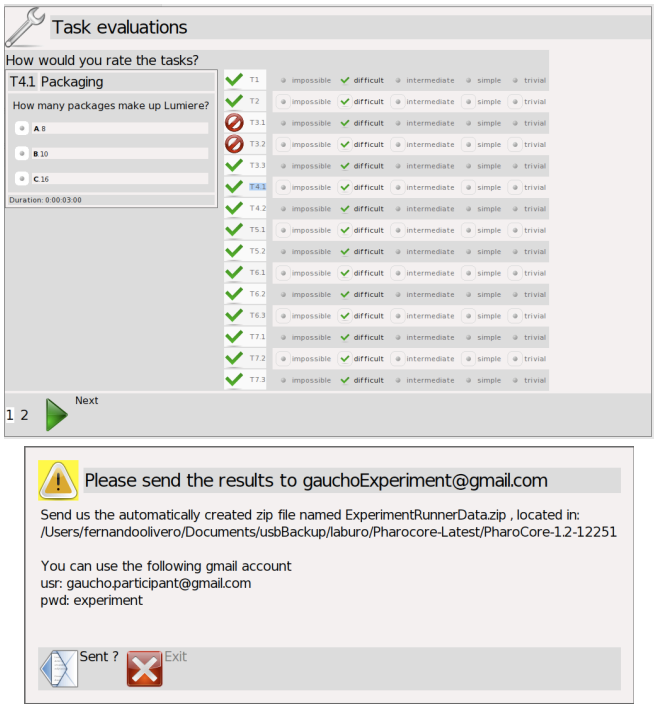

Figure 2. Biscuit output:

Related Work. To our knowledge, the only other toolset dedicated to recording fine-grained activity in controlled experiments is Emperior [7], an IDE for the Java and Groovy programming languages that records programmer navigation at the file level (Biscuit records it at the method level), records compilations of the programs, unit test runs, and periodically takes snapshots of the source code (Biscuit tracks the changes themselves); the snapshots generate an extremely large amount of redundant data. A new instance of Emperior needs to be launched for each task.

There are other tools that record usage data, such as HackyStat [8]. It collects navigation data, metrics data, and test runs. HackyStat has not been used to record controlled experiments. Similarly, Mylyn [9] records navigation and edit information as part of its activity; this data was used to evaluate Mylyn in a field study, not an experiment. Compared to these tools, Biscuit records more kinds of data (such as actual changes, not edit activity), and allows the definition of actual experimental tasks.

4. Reflections

Performing controlled experiments with human subjects is a difficult task, subject to many threats. Biscuit is an experimental toolkit aimed at reducing some of the threats related to recording and gathering experimental data. Biscuit supports the recording of answers and timing information necessary for quantitative experiments; it also collects finer-grained data—user interactions—that, on the one hand, is useful for a qualitative analysis of the data and, on the other hand, makes the experiments fully replicable.

We raise the question of whether tools such as Biscuit can improve the quality of the process by which we currently perform controlled experiments; and more importantly, if the availability of more precise and reliable data can eliminate some of the numerous threats present in controlled experiments, which are manually conducted by fallible humans.

Contrasting Gaucho's controlled experiment [6] (conducted using Biscuit) and our previous experiment [2] (conducted in the traditional manner) we noticed that:

- by relieving ourselves from bookeeping tasks, we removed ourselves as threats to validity;
- we were able to better observe the subjects during the experimental run, gathering in the process much more qualitative data about Gaucho's usability;
- data post-processing was greatly simplified;
- since the Gaucho experiment featured several very short tasks (some with a duration under a minute), we doubt that conducting the same experiment would have been possible in a "traditional" (*i.e.,* manual) way.

Conducting controlled experiments with human subjects to evaluate software engineering tools and approaches has become a necessity. Such experiments come with many threats to validity because of the humans involved; we believe Biscuit helps to remove some of the threats that regard the operation of controlled experiments.

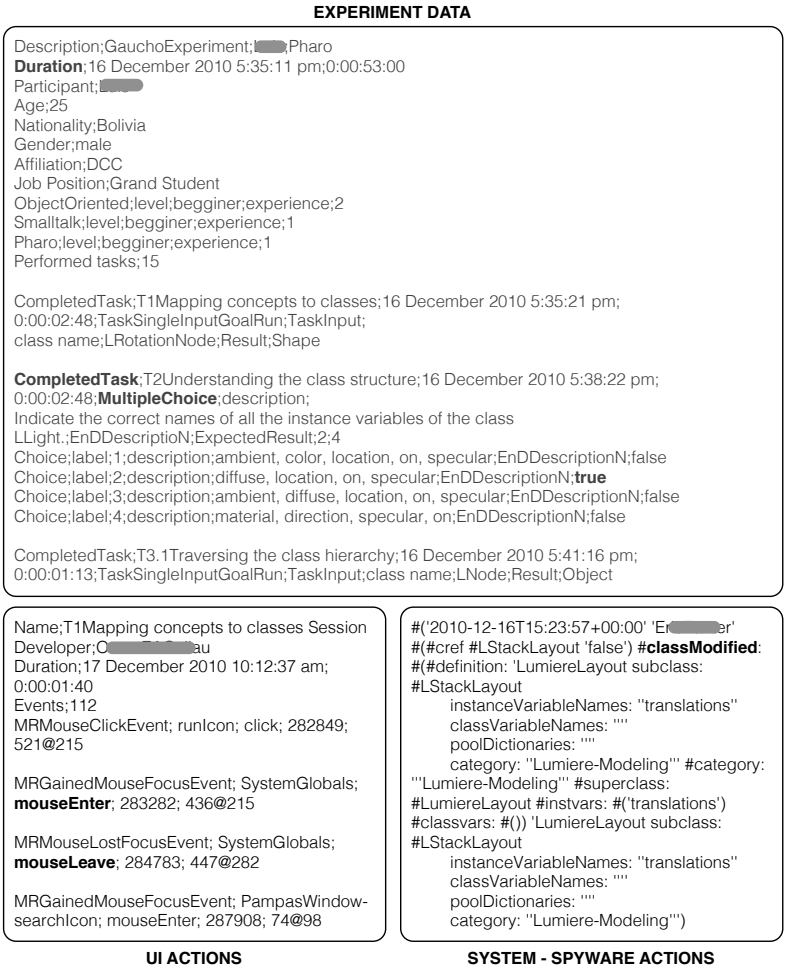

Figure 3. Biscuit output: reliable and precise data

Acknowledgments

Olivero is supported by the Swiss Science foundation (SNF Project No. 129496, "GSync"). We thank the European Smalltalk User Group (`www.esug.org`), for the sponsoring.

References

[1] B. Cornelissen, A. Zaidman, and A. van Deursen, "A controlled experiment for program comprehension through trace visualization," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, 2011.

[2] R. Wettel, M. Lanza, and R. Robbes, "Software systems as cities: A controlled experiment," in *Proceedings of ICSE 2011*. ACM Press, 2011, pp. 551 – 560.

[3] A. J. Ko, R. DeLine, and G. Venolia, "Information needs in collocated software development teams," in *Proceedings of ICSE 2007*. IEEE Press, 2007, pp. 344–353.

[4] R. Robbes and M. Lanza, "Spyware: A change-aware development toolset," in *Proceedings of ICSE 2008*. ACM Press, 2008, pp. 847–850.

[5] F. Olivero, M. Lanza, and M. Lungu, "Gaucho: From integrated development environments to direct manipulation environments," in *Proceedings of 1st Intl. Workshop on Flexible Modeling Tools*, 2010.

[6] F. Olivero, M. Lanza, M. D'Ambros, and R. Robbes, "Enabling program comprehension through a visual object-focused development environment," in *Proceedings of VL/HCC 2011*. IEEE Press, 2011.

[7] M. Steinberg, "What is the impact of static type systems on maintenance tasks? an empirical study of differences in debugging time using statically and dynamically typed languages," Master Thesis, University of Duisburg-Essen, 2011.

[8] P. M. Johnson, H. Kou, J. Agustin, C. Chan, C. A. Moore, J. Miglani, S. Zhen, and W. E. J. Doane, "Beyond the personal software process: Metrics collection and analysis for the differently disciplined," in *Proceedings of ICSE 2003*, 2003, pp. 641–646.

[9] M. Kersten and G. Murphy, "Using task context to improve programmer productivity," in *Proceedings of FSE 2006 (16th SIGSOFT Symposium on the Foundations of Software Engineering)*. ACM Press, 2006, pp. 1–11.