Erase and Rewind – Learning by Replaying Examples

Lile Hattori[1], Alberto Bacchelli[1], Michele Lanza[1], Mircea Lungu[2]
[1]REVEAL @ University of Lugano – [2]SCG @ University of Berne

Abstract

Learning is much built on observing, retaining, and replicating behavior witnessed in a model. On this basis, instructors often teach informatics by providing programming examples to be observed and analyzed by learners. However, professors usually show an example once and provide only its final version. This hinders the students' need of repeated observation and replication. To overcome this limitation, we propose Replay, an Eclipse plug-in that records every code edit of a programming session, making it available as an interactive "tape". Professors can design the steps of an example, and "play" them as live sessions in class, without the burden of coding while explaining; students have their hands in the complete code history, can observe it repeatedly, and interact with it at any moment.

1. Learning by Replaying the Examples

Professors who often write code examples in class to complement explanations are a clear evidence of the importance of observing and understanding the creation process behind the examples. Due to resource constraints, this process, however, is usually shown *once* and professors only release material in its final static form. Learners, thus, cannot observe the creation process again and might find difficult to retain it.

We devised our *Learning by Replaying* method through Replay [1], a toolkit made of two Eclipse plug-ins: (1) Replay Recorder – records a snapshot of the source code every time a file is built, generating an *executable code tape* of the programming session; (2) Replay Player – once a tape is loaded, the Player lists every step, and the learner can (1) play and watch the code evolving, (2) pause the replay, and (3) step through the items of the list.

We argue that *Learning by Replaying* helps the learning process in various aspects: Learners have access to the creation process of source code, where each step is a cue to reminding the professor's explanation, and they can dynamically interact with the code and the whole context at any stage of the development session.

References

[1] L. Hattori, M. Lungu, and M. Lanza. Replaying past changes on multi-developer projects. In *Proceedings of IWPSE 2010 (11th Intl. Workshop on Principles of Soft. Evolution)*, pages 13–22, 2010.