GrandeOmega - a didactic model for learning programming

Abstract

Teaching (and learning) how to program is an open problem. Both teachers and students cite a variety of issues, ranging from difficulty bridging the gap between theory and practice, to lack of simple assignments suitable for beginners. We believe that technology can offer a powerful tool in supporting a solution to these problems, but at a high cost in internal complexity. For this reason we have built GrandeOmega: a generic, web-based code-interpreter that can be easily used to build interactive lectures and assignments where students experiment with learning how to program while receiving real-time feedback.

1 Introduction

Learning how to program is a daunting task. There is a myriad of studies [...] documenting all sorts of didactic approaches in teaching programming in both lower and higher education by means of games, flipping the classroom, role playing, massive online courses, and much more.

Unfortunately, a clear winner has not yet emerged. The broad variety of methods is perhaps a symptom of a still open problem, which we believe is still very much present and made even more urgent by government calls to increase the number of graduates [...] to keep driving knowledge economies.

We observe that in many institutions, either the survival rates of students are quite low [...], or (in some extreme cases) the quality of the curriculum is adjusted downwards to avoid punishing students too much. Feedback from companies around the authors is often quite damning: the quality of graduate programmers is often too low, citing lack of understanding of basic aspects of programming such as memory models, concrete semantics of languages, types and type systems, with the result of slow, bug-ridden code being shipped all too often [...]. In some places we are even witnessing a slow fading of the engineering from software engineering.

We propose that learning to program is indeed fundamental, but it must happen at a high level. Those who will build the digital infrastructure of tomorrow must be able to do so with reliable, high-performance results: an infrastructure that barely works and holds together will be worthless.

Problem statement: the broader issue that we try to tackle in this paper is the study and improvement of teaching programming in higher education institutions.

To do so, we will begin by analysing the issue of how complex is it to learn how to program (Section 2). We will then provide a sketch of our solution with all the ingredients we believe to be fundamental for solving the issue (Section 3). We take the ingredients together and discuss the architecture of our actual implementation of such a solution (Section 4). Finally, we provide an evaluation of the impact of our (large) trial at the Rotterdam University of Applied Sciences (Section 5).

2 Why is it so hard?

Within the context of higher education, learning programming is hard for both beginners and students with past experience.

It is suggested by some [...] that learning programming is no different than most other complex skills: it takes roughly ten years (ten thousand hours) to become truly proficient.

The reason why it takes so long is disarmingly simple. Programming requires both the ability to **understand** and to **design** code.