Studies into interactive didactic approaches for learning software design using UML

Stikkolorum, D.R.

Summary

Software design is a challenging task for students. Due to the trend of increasingly complex systems, teaching students software design becomes also challenging. It is known that students struggle with the (abstract) task of software design, therefore new teaching approaches are necessary.

This dissertation explores different ways of improving and extending our current approaches in software design education. The main objective of this research is:

Main Objective *How can we improve the ways of teaching software design?*

We studied different aspects of the main objective: measuring software design skills; guidance and feedback on design tasks; and, student motivation and engagement in software design activities.

We explored interventions through the use of tools we developed for assessing design skills, modelling, guiding and grading. We centered our approach around design principles. Design principles form the foundations of the complex balancing of multiple conflicting objectives that good software designers should eventually master.

Next, we describe our different interventions and developed tools.

We designed an on-line multiple-choice test as an instrument for measuring software design skills and abstract reasoning skills of students. Our research revealed the relationship between abstract reasoning and the ability of solving software design problems. We see opportunities in exploring educational interventions for specific (abstract) reasoning tasks and/or alternative teaching methods.

Lecturers can use our test to diagnose students and choose the appropriate interven-

tions.

For motivating students for learning software design and for keeping them engaged, we explored an educational puzzle game.

By practising making design decisions, the players learn about balancing between the different software design objectives. An important instrument in the game is the visual feedback system that guides students towards their solution. We see the game as an opportunity to support students' self-assessment possibilities by using the built-in feedback mechanism. The interactive game keeps them engaged.

Because of the need for a UML tool that could be used as an educational tool as well as a research tool we developed WebUML. The web-based architecture combined with a logging system enabled larger scale classroom studies that could be analysed with statistical analyses. We were able to collect data about student activities during class diagram analysis and design. Our tools open up new ways of studying the process of software design. Having the possibility to analyse students' modelling behaviour enables us to develop better educational programs and tools. Recognising certain strategies of making designs can help us to give better feedback during a design-task rather than after completion of the task.

By analysing the logs of students using the WebUML tool when making designs, we found typical strategies students use to make class designs: Depth Less, Depth First, Breadth First, Ad Hoc. The strategies differ in whether a student first focuses on high-level class structure or focuses on individual class detail.

As another teaching-intervention, we analysed peer-reflections of students during an analysis and design assignment. The peer-reflections revealed a collection of challenges the students face. We identified common problems and difficulties and presented 10 concrete points of concern for teaching software design. From the recorded peer-reflections we conclude that peer-reflection helps students to express their difficulties in modelling software designs. The students actively discussed the difficulties that they had as well as the progress and quality aspects of their models. With the presented points of concern we expand the knowledge about common difficulties in students' learning of software design.

For another approach of distilling the struggles that students face while making a software design we observed the interaction between teaching assistants (TAs) and students during an analysis and design course. We were able to distil and categorise the common challenges students have recorded where TAs supported them. Further, we have an overview of the typical interventions TAs use when helping students during their assignment tasks.

We explored the application of automated feedback and grading of UML class diagrams in order to enable students to practice more often and reflect on their own progress by self-assessment. In addition to the evaluation function of such mechanisms the feedback enables the guidance of students towards their solution. The feedback is about: using the right semantics, such as the appropriate translation of a relationship between concepts and satisfying the requirements of the assignment, such as identifying the relevant concepts within a given context.

We collected the data of submitted software design assignments of bachelor students. On this data we applied machine learning for trying to grade assignments. Based on our results we conclude that using machine learning for a classification that uses the same precision as a 10 point grading[2] scale is not accurate. However, the current prediction model does give users a rough indication of the quality of their models.

We presented our pedagogical feedback agent as an extension to WebUML. This agent gives feedback to students while they are learning to make UML class designs.

The students appreciate the guiding role of the feedback agent: the way it confirms steps that were done in the right way and suggestions for next steps.

Next to tooling, we proposed a workshop that integrates UML modelling into agile development. An integrated software development approach helps students to understand the role of (UML) modelling in a software development cycle. In addition we are able to collect information about students' difficulties that arise during discussing the modelling task in a group. Students have a positive judgement towards the workshop. The students judge that they have improved their modelling skills and that the workshop gave insight into the relationship between modelling and the software development process.

With the above-mentioned studies we expanded the repertoire of educational interventions and tools for learning software design. With proper feedback these interventions and tools guide the students during their design tasks. We invite other lecturers to use and build on our approaches in order to extend our insights and enrich future interactive learning environments.

[2] This is the conventional scale for grading exams in The Netherlands.