Improving Teaching in an Introductory Programming Course: Visualization Tools and Criteria-based Assessment

Course context and problems' description

The TND012 course is the first programming course offered to three different engineering programmes, MT, ED and KTS, during the first year (HT1). No previous knowledge of programming is required and the programming language used in the course is C++.

As course responsible, I realized that the course had a number of aspects that needed to be improved.

The final examination, consisting of a computer exam, was initially mostly a quantitative approach to assessment. More concretely, each question awarded a number of points and a minimum threshold was established for each of the three course grades (e.g. at least 50% of the total exam points was required for grade 3). Although this might seem an objective and precise way of grading the exam, consequently easy to motivate to the students, it had the drawback of conveying to the students the idea that all they needed was to scrabble points where ever they could to reach a given grade. This leads to the first problem.

- **Problem 1**: better alignment of assessment and course goals.

Feedback is important to both agents involved in any course: students and teachers. If teachers strive to give feedback to their students about how well students are getting to the course goals, it then turns out that teachers themselves get a better understanding of their teaching performance. The tools being used in TND012 to give students feedback about their understanding of course topics and capability to solve problems were mostly based on the interaction between lab assistants and students, during lab sessions. Although the feedback obtained during the lab sessions is seen as valuable, students pointed out that they were not getting enough continuous feedback during the course. This request is expressed clearly is the students’ course evaluations and, for instance, in the LinTek NSI report\(^1\) from 2010. In addition, the course had no tool that provided individual feedback to the students, with exception of the final exam. Feedback given in the lab sessions is directed to groups of two or three students. In a beginners’ course, like TND012, this may be a serious drawback. This leads to the second problem.

- **Problem 2**: improve the feedback mechanisms of the course.

The level of abstraction required by programming is one of the major challenges for beginner students. For instance, the programmers have to be able decompose the initial tasks and then map this plan into the constructs of the programming language. This is often perceived as no trivial steps because, for instance, people can be quite unable to say how they perform certain tasks. Having a library of practical examples enhanced through visualization would certainly help students in learning. Although there are such visualization tools for more advanced courses, such

---

\(^1\) LinTeK NSI report from 2010 is available at [http://www.lintek.liu.se/assets/Om-LinTek/Dokument/linteks-rapport-om-nsi-2010-slutgiltig.pdf](http://www.lintek.liu.se/assets/Om-LinTek/Dokument/linteks-rapport-om-nsi-2010-slutgiltig.pdf) (see pag. 10)
as data structures, the use of visual tools in teaching programming concepts to novices is rather limited. This leads to the third problem.

- **Problem 3:** improve understanding of important abstract programming concepts and stimulate students’ interest for programming.

Programming courses are essentially practical and students need to work during the entire course to gain the needed experience to build programs, and therefore, attain the course goals. However, there was a noticeable tendency for first year students to put time for the course when the final exam date approached. One possible explanation for this behavior could be that math courses are a major competitor for the students’ time and attention, during the first year at the university. Consequently, it was needed to find a way to engage students’ efforts earlier in the course. This leads to the fourth problem.

- **Problem 4:** keep students actively working with the course during the entire course.

**PUG-project goals**

In the context of the pedagogic development of courses supported by LiU, a small project was formulated as part of the process of tackling the problems described above. The project goals are described below.

- To redesign course TND012 assessment such that a clearer criteria-based examination is used. In this way, we aim at tackling problem 1 above.
- To design a set of home assignments, to be delivered during the course and before the final computer examination. The home assignments can help in addressing problems 2 and 4.
- To develop a library of visualization tools for basic programming concepts. This part of the project is preferable developed as thesis work, for a Media Technology student. Our purpose is to address problem 3 described above.

**Implementation and results**

The SOLO-taxonomy, adopted by LiTH for describing different levels of increasing complexity in a student’s understanding of a subject, was my starting point for designing the individual assessments (final computer exam and home assignments) in the course and their grading system. However, this taxonomy is too technical and abstract to be easily grasped by students. Consequently, I adopted a different system and then established a connection with the SOLO-taxonomy. This system has three levels, directly connected to the three courses final grades, and briefly described below. The course goals, as listed in the *studiehandbok*, are presented in appendix.

- **Minimal level**, corresponding to grade 3. At this level, students must show they have attained the learning outcomes 1-8. Moreover, they should be able to solve basic problems and structure simple programs. This level corresponds mostly to the uni-structural and multi-structural level of the SOLO-taxonomy, since the students should be able to write programs focusing in a certain aspect or few aspects easily connected.
- **Medium level**, corresponding to grade 4. At this level, students must show they have also attained the learning outcome 9. Thus, students should also be able to structure medium-
complex programs integrating coherently different programming concepts. For instance, structuring a program with the help of functions, using different methods of passing information to functions. This level corresponds to the relational level of the SOLO-taxonomy.

- **Beyond medium level**, corresponding to grade 5. At this level, students must have attained all the learning outcomes and, additionally, be able to tackle more complex practical problems displaying creativity, deeper knowledge, and capacity to integrate different techniques. This level corresponds to the extended abstract level of the SOLO-taxonomy.

The current form of the final exam has three parts, reflecting directly the three levels described above: Part I is at the minimal level and awards grade 3; Part II is at the medium level and awards grade 4; Part III is beyond medium level and awards grade 5. Part I of the exam is mandatory, while students can choose whether to write Part II and/or Part III. To avoid having a sudden shift from a quantitative assessment to a holistic judgment, which may be perceived by students as more subjective, each exam exercise still awards a number of points. However, a minimal threshold is imposed separately on each part of the exam.

Final course evaluations performed by the students show that students understand the connection between the structure of the exam and the course goals, as the following statement extracted from course evaluation, done in 2013, shows: “Tentan upplevs som bra utformad efter kursens mål.”. A positive side effect of the new final exam is that it also became easier to motivate the exam grade to students, based on which learning outcomes have, or have not, been attained. Students who wish to improve their grade (by doing an upcoming exam of the course) also know on which course goals they need to focus.

A set of individual home assignments consisting of a number of programming exercises, have been designed. Students have a couple of days to submit their solutions. After the deadline, teachers involved in the course look over the students’ solutions and write a short feedback report for each student.

Students find the home assignments a valuable feedback tool in the course, as the following statements extracted from the course evaluations of 2012 and 2013 show: “Duggorna var väldigt uppskattade men kan förbättras” and “Duggorna och laborationsuppgifterna var också uppskattade”. Students who attended the course in 2012 have also requested to have home assignments in the follow up C++ course, TNG033.

A consequence of the continuous feedback, obtained by doing the home assignments, is that the students work more actively with the course during the entire course. This is an essential condition for beginners’ success in a programming course. Another indication that the students are achieving the intended learning outcomes more effectively is that the approval rate on the first exam has been increasing since 2012. Before 2012, the course had an approval rate on the first exam of about 55%, while in 2013 the approval rate was of about 75%.

The description of the requirements for a visualization tool (or set of tools), to be used by the course students to enhance their understanding of important programming concepts, has been formulated has part of a bachelor thesis proposal for MT3 students, in the context of the TNM094 course. However, the number of interesting projects proposed in the context of the TNM094 course has been quite larger than the number of student groups and, unfortunately, the project I
have proposed has not been selected by any student group, yet. Thus, the third goal of the PUG-project has not been fully concretized.

**Conclusions**

The PUG-project underlying this report has the following goals. Firstly, redesign the course assessment in order to get a better alignment with the course goals. Secondly, develop an extra feedback tool in the form of a set of home assignments. Thirdly, build a visualization tool to help beginner students to understand and apply important abstract programming concepts and techniques. As explained above, this visualization tool has not yet been implemented, although has been part of a bachelor project proposal for MT3.

The first two goals of the project have been fully achieved. Nevertheless, we see the need for further improvements. For instance, students have pointed out that the feedback they obtain for the home assignments can be very heterogeneous in quality. This may be possibly explained by the fact that there are several teachers involved in giving feedback to the students. One possible way to tackle this problem in the future is to develop a library of template feedback comments that can be commonly used by the course teachers.
Appendix: TND012 course goals

1. To use built-in data types for integers, floating point numbers, text strings, arrays and records.
2. To use control structures for selection and iteration in programs.
3. To use the functionality from some basic standard C++ libraries.
4. To construct sub programs and use different types of parameters and return values.
5. To understand how to pass information between sub programs.
6. To find and correct errors, logical and syntax, in a program.
7. To develop algorithms for a problem.
8. To construct program from a problem description using different data types and control structures.
9. To construct programs modularly from subprograms, where every sub program performs one task.
10. To write programs with basic file processing.
11. To describe the following concepts: compilation, source code, object code, executable code, identifier, parameters, scope, visibility, call by value call by reference, syntax, syntax error, logical errors.
12. To describe the output of a program from the source code.