Putting Constructive Alignment to Work: A Hands-on Experience with a First-Year Programming Course

Musard Balliu
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology

Abstract

This report discusses our experience with using Constructive Alignment (CA) for developing a Programming course for first-year Computer Science students. Starting from an historical perspective, we discuss the main challenges and proposed changes towards aligning different course activities, and evaluate the corresponding outcomes in terms of student satisfaction and examination results. We also highlight new challenges and difficulties that we plan to address in the next run of the course. Based on this experience, we conclude that CA is an excellent theory for achieving student-oriented learning.

1 Introduction

In this section, we provide a high-level overview of the project. We first describe the context and identify the main issues, and then present the project idea and goals.

Background Programming is one of the very first courses which is taught to first year bachelor students in the Software Engineering and Management programme at Gothenburg University. The objective of the course is to introduce students with programming concepts, with special focus on object-oriented programming and the Java programming language. The teaching process consists of class lectures (6 hours/week) and lab sessions (6 hours/week), while examination is carried out through programming assignments and a final written exam. Besides the underlying programming concepts, the course is very much hands-on, hence practical exercises and learn-by-doing are important aspect of the course. In general, the course benefits from about 100 students and 10 teaching assistants (TAs).

Historical data The Programming course had been running for a number of years before we were assigned to teach it for the first time. Both quantitative and qualitative indicators were not very encouraging, and the previous teachers and the program director were aware of these issues. The pass rate in the previous instances of the course had been quite low. The course benefited from very good teaching material and lecture slides, and frontal lectures had an interesting combination of theory and on-the-fly live programming. The lab sessions were generally handled by teaching assistants (second/third year bachelor students) who had taken the course in the previous years. Students were generally expected to go to the lab sessions, ask questions to the TAs and work on their programming assignments. The participation of students in the lab sessions was very low. The programming assignments were graded offline by the TAs and the feedback was also provided offline. In addition, the learning outcomes were very ambitious and demanding for the time allocated to the course (less than two months).

Problem analysis and definition In collaboration with the program director and the previous course instructor, we went through a detailed analysis of the course and identified a number of issues that required immediate intervention. This was done by analyzing previous course evaluations, talking to students and leveraging the experience of the course responsible.

First and foremost, we realized that the learning outcomes were high for a very first programming course which was taught for less than two months. The requirements put high expectations on students' independent work, which resulted in a very high workload for the students. Second, the class was very heterogeneous in terms of prior knowledge and skills. Almost half of the students had no prior programming experience and many of them lacked the mathematical background that is usually required for a programming course. This had made it quite challenging to determine the right pace of the course, get accurate feedback and identify possible issues that could arise during the course. Many students had difficulties with basic problem solving tasks, which made their learning experience even more challenging. Third, lectures followed a largely traditional format, which made it challenging for students to concentrate for the whole 3-hour lecture. Fourth, we found out that it was necessary for the teacher to always be present in the lab sessions in order to propose programming exercises and monitor the progress of the students, in addition to the precious help from the TAs. Since home assignments were mandatory, students would put most of their effort to the programming assignments, once they were posted online, and ignored doing the exercises. Finally, the offline grading of the assignments was considered problematic, as the students had no interaction with the TAs during the grading process.

Idea in a nutshell This project seeks to provide a better alignment between different course activities, including lectures, lab sessions and home assignments, to maximize the fulfillment of learning objectives of the course. To this end, we investigate different activities that allow students to engage in interaction with each other and leverage elements of active learning to achieve the course objectives. As a result, teachers will receive early feedback on possible issues with the course and act accordingly to solve them. The pedagogical idea is based on the theory of *constructive alignment*, which provides a principled way of aligning the learning outcomes (what the students are supposed to understand, be able to do and be able to relate to at the end of the course), teaching and learning activities (the different activities taking place during the course, i.e., lectures, lab sessions, assignments and examination), and the formal examination.

In a nutshell, we propose the following activities and changes to the course: (a) work on student motivation and enthusiasm; (b) anonymous online quizzes to check students' progress; (c) training of problem solving capabilities; (d) quick recaps at the beginning of each lecture (e) lab exercises to help students completing the graded home assignments; (f) enforce work-in-group and student presentations of the home assignments, and (g) revision of the learning outcomes as to make them more efficient towards student learning.

Goals The main goal of this project is to apply the theory of constructive alignment at the letter, providing a student-oriented learning environment and increasing the qualitative and quantitative indicators of the course. To this end, we set out to achieve the following goals:

- Enhance students' experience by introducing elements of active learning
- Help teachers to identify possible issues and get feedback early on
- Provide a better alignment of course activities towards the fulfillment of learning objectives
- Increase cooperation between peers and train presentation skills
- Make the learning experience more productive, interactive and fun
- Increase the course quality by improving on quantitative and quantitative indicators

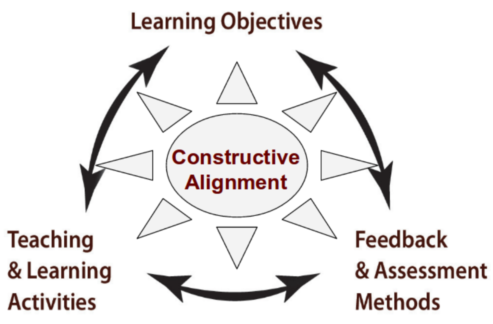
2 Methods and Results

In this section, we give a brief introduction of the pedagogical theory that we use in the project, and provide a detailed explanation of how such theory was implemented for our course. Finally, we conclude by discussing the results.

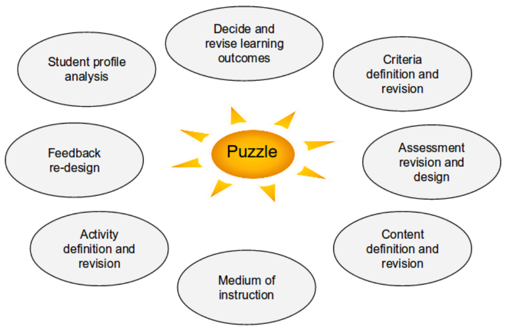
2.1 Constructive Alignment

Learning theories are systematic conceptual frameworks that explain the how the students learn, namely how knowledge is absorbed, processed and retained during the learning process. In general, cognitive, emotional, and environmental influences, as well as prior experience, all play an important role in how understanding is acquired or changed, and knowledge and skills are retained [9]. Learning theories comprise a number of teaching philosophies that range from teacher-focused approaches to student-focused approaches [11]. Constructivism preaches that the ability to learn relies to a large extent on what the learner already knows and understands, and the acquisition of knowledge should be an individually tailored process of construction. In this project we investigate the theory of the constructive alignment [3, 4, 2, 12] as depicted in Figure 1. In essence, CA states that the best way of creating knowledge is by means of activities that students engage in, rather than by direct knowledge transfer from the teacher to the student. The role of the teacher is seen as a facilitator that produces teaching activities and creates an environment that enables students to engage in deep learning.

To this end, CA provides a aligned system of instruction, a *web of consistency*, where students are "forced" to engage in appropriate learning activities. Figure 1a describes the main components of the theory of constructive alignment. By putting the student activities on the spotlight, the teacher has to be clear about the learning objectives and state clearly what the students should learn. Clear learning objectives should drive teaching and learning activities, namely activities that encourage students to pursue learning in a way that is likely to achieve the objectives. Furthermore, clear objectives should also drive feedback and assessment methods, namely methods that allow the teacher to assess how well the students have learned and what feedback needs to be provided. Figure 1b provides a fine-grained representation of the constructive alignment ingredients which we have fully applied to designing a programming course.



(a) Idea


Decide and revise learning outcomes
Student profile analysis
Criteria definition and revision
Feedback re-design
Puzzle
Assessment revision and design
Activity definition and revision
Content definition and revision
Medium of instruction

(b) Activities

Figure 1: Constructive Alignment

2.2 Putting Constructive Alignment to Work

The constructive alignment sun in Figure 1b defines the building blocks for our application of the theory to a concrete scenario: *the use of CA for developing a programming course*. An excellent step by step guide on designing a course for constructive alignment can be found in [12].

2.2.1 Learning Outcome Revision

In constructive alignment, learning outcomes are a key player in defining course activities, assessment and feedback. They provide a clear statement of what a student is expected to be able to do at the end of a period of learning, which can be a course, a session, an assignment or any other activity. Learning outcomes describe and quantify the minimum achievement, i.e., the goals, a student needs to demonstrate in order to pass the course [1, 4].

Based on the data from past experience, we decide to revise the learning outcomes at multiple levels. The course syllabus contained topics that were considered too advanced for a first year programming course, and for the short duration of the course itself. Moreover, some of the outcomes overlapped with courses running in parallel with our course. The first revision of the learning outcomes was done at the course level by making them more realistic and reducing overlaps with other courses. This change enabled a new structure for the entire programme and yielded a new basic course for training problem solving skills. We propagated the changes to the level of class lectures, exercise sessions and home assignments. Interestingly, the changes did not affect the final examination as the removed material was never assessed in the course. This gave us more confidence that the revision was indeed needed. We also revised the learning outcomes of the exercise sessions and home assignments by essentially aligning the two. This increased students' participation in the exercise sessions as it would allow them to train their programming skills and get the required knowledge for solving the home assignments. The revision of the intended learning outcome at the course level is reported in Appendix **??**.

2.2.2 Student Profile Analysis

At the beginning of the course, we performed a student profile analysis with the aim of identifying prior knowledge and expectations, and clarifying what we expected from the students in order to succeed in the course. Based on the past experience, we were well aware of the gap in prior knowledge between students. To this end, we implemented the following changes: (i) We chose a course book that follows a problem solving approach to introducing programming concepts. This allowed the students to catch up with the basic mathematical and logical reasoning that is needed for a programming course. (ii) We introduced exercises that were aimed at training such skills, both during the lectures and exercise sessions. (iii) We decided to have regular meetings with course representatives in order to get feedback about the actual effects of these changes.

Moreover, a very important factor to success was the motivation and enthusiasm of the teacher and TAs in introducing the topics to the students. Programming is a very important course that affects the entire student experience at the university, as many other courses depend on it strongly. We worked hard to motivate the students and make them like programming from the very beginning. This was done by encouraging and helping students whenever they experienced failures, and by introducing cool applications that would get them excited about programming.

Finally, we created a safe and student-friendly learning environment based on collaboration and inclusion, which encouraged students to ask questions and be active during the course activities. We did this by having small chats during the breaks, stimulating interaction and discussion, and, importantly, setting out the challenge of learning students' names. This last choice allowed to break up the ice between the students and the teacher, thus getting everyone aboard towards the common goal: learning how to program.

2.2.3 Content and Resource Revision

The first learning experience at the university level can be intimidating for many students. To facilitate the first impact, special care was dedicated to information dissemination. We put additional effort on setting up a simple and well structured course web page, which made it easier for students to find course-related information. This effort was highly appreciated and acknowledged by many students in the course evaluation. Moreover, the course benefited from 10 TAs, who participated to the exercise sessions and helped students with exercises and home assignments. We decided to have a common email address for the course, accessible to the TAs and the teacher, where students could send in their questions and get fast feedback. In addition, TAs leveraged social media like Slack and Facebook to communicate with the students and answer their questions.

2.2.4 Revision of Activities

To properly map the course activities to the learning outcomes, we introduced several new activities. First, we leveraged elements of active learning to increase students' interaction during the class lectures. The main new activity were anonymous online quizzes through the *Kahoot* system. We proposed quizzes after the completion of every milestone of the course. Quizzes provided an interactive and fun environment during the lectures, but more importantly, they allowed students and the teacher to get a general impression of the main difficulties, for then trying to explain them better in the following lectures. Furthermore, we introduced quick recaps at the beginning of each lecture in order to connect everything to the main picture and present the learning outcomes for the given lecture. Together with live coding and problem solving sessions, these changes were highly appreciated by the students. We made large use of the whiteboard to explain several threshold concepts [8, 7] like *references, arrays, call by value semantics, static and instance variables/methods.* This turned out to be a crucial choice that helped students understand the main concepts way even before getting into the nitty-gritty programming features. We established regular meetings with the TAs to discuss the set of exercises before each session, and define common guidelines for the evaluation of programming assignments. We strongly encouraged students to work in groups of two, and enforced the requirement of presenting home assignments to the TAs. This choice increased collaboration between students and trained their presentation skills.

2.2.5 Medium of Instruction

As mentioned earlier, the course benefited from a large array of instruction mediums including web platforms, social media, whiteboard, email and integrated development environments for programming, e.g., Eclipse IDE.

2.2.6 Assessment and Feedback

We sought a combination of formal and informal assessment methods and increased the number of feedback channels [5, 10, 6]. During the lectures, quizzes became an excellent tool both for the students and the teacher to get feedback about their progress. Furthermore, exercise sessions provided another channel for feedback exchange. Oftentimes, we organized small informal competitions between students to solve programming exercises. At the end of each session, we published sample solutions that would allow students to compare their solution with ours. The informal nature of such activities created a great learning environment, without the pressure of formal examination. We chose the exercises in order to cover the intended learning outcomes of the given session, but also to help students solving the graded assignments. This enabled a smooth transition between informal to formal assessment, thus making the assignment grading an pleasant opportunity for students to show off their programming capabilities. The group presentation of the assignments created another feedback channel where students could improve their presentation skills and get immediate feedback. We did not change the written exam as we were confident that it was already covering the majority of the learning outcomes that we wanted to examine. Finally, we had a revision session to allow students discuss their exam performance with the teacher.

2.3 Results

After analyzing the student evaluations and the examination results, we found out that the constructive alignment of the programming course produced excellent results. 92% of the students evaluated the course as good and excellent, while 70% of the students managed to pass the course after the first examination. In the evaluation, we asked specific questions about the proposed changes and activities and they were highly appreciated by the large majority of the students (over 85%). Many students felt enthusiastic about programming and all the activities in the course. Here we report some comments from the students' perspective:

- "The teacher was so caring in the matter of our learning and he was really focused on the study goals."
- "The teacher was always there to answer my questions. The TAs were very helpful doing a remarkable job (I will bring the idea of having TAs to my country)!"

- ”The teaching was very interactive and easy to follow and keep up with. Everything was explained at a level fit for both beginners and people with some experience. It was by far the most fun and fulfilling course I’ve ever had.”
- ”I simply love the exercise sessions and the assignments! Great way to keep the engagement levels up and good practice for the exam.”
- ”The Kahoot questionnaires were quite a nice start of the wednesday lectures as well as a fun way to test the progress. The course structure was good and should stay relatively the same. ”
- ”Almost everything. The TA sessions. Magic to have all that support - the best teacher/coach. The ”we love programming” approach is really inspiring. Very good to have one lecture and repeat it next time.”

On the other hand, we received several excellent suggestions that we will address for the next run of the course.

- ”The course was going a bit fast so that put us in so much pressure trying to catch up with everything.”
- ”The kahoot! quizzes! They were a lovely medium, but I’d have loved a little introduction to the medium prior.”
- ”The format is good. The only issue with some students is that working in crowded places is not their thing, thus they found another less crowded room or they skipped the lab altogether, working from home.”
- ”The learning curve should be smoothed out and easier tasks should be given first and the difficulty then ramped up as time progresses.”
- ”Make a longer period for the course!”

References

[1] Anderson, L.W., Krathwohl, D.R., Airasian, P., Cruikshank, K., Mayer, R., Pintrich, P., Raths, J., Wittrock, M.: A taxonomy for learning, teaching and assessing: A revision of bloom’s taxonomy. New York. Longman Publishing. Artz, AF, & Armour-Thomas, E.(1992). pp. 137–175 (2001)

[2] Bain, K.: What the best college teachers do. Harvard University Press (2011)

[3] Biggs, J.: What the student does: Teaching for enhanced learning. Higher education research & development 18(1), 57–75 (1999)

[4] Biggs, J., Tang, C.: Teaching for quality learning at university (society for research into higher education) (2007)

[5] Gibbs, G.: Using assessment strategically to change the way students learn. Assessment matters in higher education 41 (1999)

[6] Kember, D.: Action learning and action research: Improving the quality of teaching and learning. Psychology Press (2000)

[7] Land, R., Cousin, G., Meyer, J.H., Davies, P.: Threshold concepts and troublesome knowledge (3): implications for course design and evaluation. Improving student learning diversity and inclusivity 4, 53–64 (2005)

[8] Meyer, J.H., Land, R.: Threshold concepts and troublesome knowledge. Overcoming barriers to student understanding: Threshold concepts and troublesome knowledge pp. 3–18 (2006)

[9] Ormrod, J.E.: Human Learning: Pearson New International Edition. Pearson Higher Ed (2013)

[10] Sadler, D.R.: Formative assessment and the design of instructional systems. Instructional science 18(2), 119–144 (1989)

[11] Trigwell, K.: Judging university teaching. International Journal for Academic Development 6(1), 65–73 (2001)

[12] Weurlander, M.: Designing a course for meaningful learning. A step by step guide. CME guide (1) (2006)