2016 IEEE/ACM 38th IEEE International Conference on Software Engineering Companion

Reflections on Applying Constructive Alignment with Formative Feedback for Teaching Introductory Programming and Software Architecture

Andrew Cain
Department of Computer Science and Software Engineering
Swinburne University of Technology
Hawthorn, Australia
acain@swin.edu.au

Muhammad Ali Babar
School of Computer Science
The University of Adelaide
Adelaide, Australia
ali.babar@adelaide.edu.au

ABSTRACT

Constructive alignment is a student-centred approach to teaching and learning that aims to enhance student learning through a combination of constructivist learning theories and aligned curriculum. This paper presents two case studies where units have been developed to apply the principles of constructive alignment in the area of computer science and software engineering. It outlines the role of formative feedback and delayed summative assessment as a means of embedding constructivist learning theories in the application of constructive alignment. The discussion outlines some of the challenges and advantages gained from the greater focus on formative feedback during the teaching period, and presents some recommendations for others considering applying constructive alignment.

Keywords

Constructive Alignment, Case Study, Assessment

1. INTRODUCTION

Constructive Alignment, originally proposed by Biggs [5], is an approach to teaching and learning centred on the philosophy that "it is what the student does that counts". The principles of constructive alignment combine constructivist learning theories and aligned curriculum, with the aim of enhancing learning outcomes by encouraging students to adopt deep approaches to learning.

Teaching and learning in higher education has been greatly influenced by constructive alignment. The effect of this can be seen in articles such as [19], which stated that incorporating the principles of constructive alignment was one of the key aspects for successfully writing for the Journal of Higher Education in Geography. Similarly, key principles from constructive alignment have also been central in wider changes, such as the adoption of outcome-based education and aligned curriculum used to underpin Australia's education system in the Australian Qualifications Framework [15]. The importance of constructive alignment for computer science and software engineering has been highlighted by Armarego [3], who concluded that:

> "...it is no longer appropriate to rely on traditional teaching as the basis for the learning process – these methods do not align well with the requirements of the profession, and inhibit many (actual as well as potential) students from engaging with the discipline." [3]

It can be suggested, therefore, that sharing case studies that demonstrate effective means of applying constructive alignment will help advance teaching within the fields of computer science and software engineering.

This paper presents two different applications of constructive alignment: one related to teaching introductory programming to first year undergraduate students, the other being an introduction to software architecture taken by postgraduate students. The approaches differ in their assessment strategies, but both predominantly use formative feedback during the teaching period and delay summative assessment until the end of the unit's delivery. Section 2 introduces relevant background on the related educational theories used to underpin the approaches presented. This is followed by an outline of the two approaches in Section 3 and Section 4, with discussions of the implications from these case studies in Section 5. The paper then outlines some concluding remarks in Section 6.

2. BACKGROUND

Constructive alignment, as proposed by Biggs [5], is a model of teaching that combines constructivist learning theory and aligned instruction design with the goal of improving learning for all students. Biggs' model, with its general motto of "it's what the student does that counts", is a student-centred approach to education in which assessment, and teaching and learning activities are clearly and intentionally aligned to unit learning outcomes. The model is centred upon the student, who builds knowledge as a result of their interaction with the teaching and learning activities and assessment; an idea derived from constructivist learning theories. This view of the learning is paired with aligned curriculum that has its foundation in instructional design literature. To encourage students to engage effective approaches

ICSE '16 Companion, May 14-22, 2016, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/2889160.2889185

to learning, activities and assessments are aligned to unit learning outcomes that are stated at appropriate cognitive levels.

2.1 Constructivist Learning Theories

Constructivism is a theory to explain the development of knowledge, which focuses on the active role of the learner in constructing their own understanding with origins dating back to Piaget [28].Constructivism exists in several forms: cognitive, individual, post-modern, radical and social constructivism [26, 34], and each form of constructivism has various implications for teaching and learning.

Constructivist epistemology holds that knowledge is a human construction that is informed by observation of, and constrained by, external reality. The implications for this, in terms of education, is that constructivism centres teaching around the activity of guiding the learner's active construction of knowledge, with the objective of helping the learner think and act like an expert [20, 38]. When guided by this epistemology, assessment involves qualitatively evaluating the structure of the learner's knowledge in relation to the structure of an expert's knowledge.

Teaching using constructivist based approaches requires educators to adopt a theory of teaching and learning that centres upon guidance of student learning, rather than on transmission of unit content. A study of university teaching staff [21] examined relationships between how staff intended to present a topic and how it was subsequently delivered. The results indicated that where staff viewed the topic as "knowledge as given", they adopted a teacher-focused approach using information transmission forms of delivery that are inconsistent with constructivist theories. These strategies may result in students perceiving memorisation and recall as being what is required to succeed in the unit, and therefore adopt surface approaches to learning [23, 24]. In contrast, student-focused approaches were taken where staff viewed the object of study as related to conceptual change. These approaches were more consistent with constructivist thinking, and learning tended to focus on higher cognitive level activities, with a broader focus on the discipline, practice or life-long-learning, and encourage students to engage in deep approaches to learning.

Constructivism is often promoted by educators, but the associated teaching and assessment practices have not transitioned to common education practice [27]: a symptom of the disconnect between educators' espoused theory and their theory-in-use [2]. To successfully implement constructive alignment it is, therefore, important to identify and adopt the key aspects from constructivism, as outlined by Biggs [5, 6, 8].

Biggs reason for adopting constructivism as a central philosophy was due to its emphasis on the students active role in constructing their own knowledge [5]. This suggests a pragmatic use of constructivist learning theories, avoiding some of the criticisms aimed at these theories when "pursued to unproductive extremes" [1]. This means the adoption of views central to all forms of constructivism:

- Knowledge is constructed, not transmitted via communication alone.
- Teaching involves creating a context in which learners are able to construct appropriate cognitive models through individual and social activities.
- Errors in understanding are opportunities for further learning, as these help indicate the students' current level of development and can be used to guide future learning activities.

2.2 Aligned Curriculum

Curriculum alignment was defined by Tyler [37] as occurring when material learnt in earlier years is built upon and supported by subsequent classes. The model of alignment presented by Tyler consisted of four steps: 1) identifying objectives, 2) choosing appropriate learning experiences, 3) organising these experiences to maximise learning, and 4) evaluating the results of learning. This provided alignment between the objectives, teaching and learning activities, and assessment tasks. This model was further refined by Ramsden [30], who indicated that the alignment can be achieved as long as assessment matches the learning objectives; as from the students' perspective the curriculum is always defined by the assessment. The benefits of instructional alignment were examined by Cohen [14] who reported that students achieved significantly better results when the teaching was aligned to the assessment, than in cases where instruction was not aligned.

Early work on approaches to learning [22] indicated that the major influencing factor had been the students interpretation of what was expected of them. With constructive alignment, Biggs proposed the use of aligned curriculum with learning outcomes expressed at clearly distinct cognitive levels to better guide students to use appropriate cognitive activities [5]. This built on Biggs earlier work on the SOLO Taxonomy [7], which defined five structural levels of observed outcomes: pre-structural, uni-structural, multi-structural, relational and extended abstract. Each of these levels can then be associated with verbs that are likely to elicit cognitive activity at that level, and then used to define a unit's learning outcomes. Students then work toward achieving and demonstrating these learning outcomes, have a clearer expectation of the kinds of cognitive activities that are expected of them.

2.3 Formative Feedback

In education, assessment serves one of two purposes [32, 10]: supporting learning, or evaluating outcomes. These two forms of assessment are known as *formative assessment* and *summative assessment*. Formative assessment aims to assess student learning for the purpose of providing *feedback* that the student can use to improve their understanding. This is distinct from summative assessment which aims to assess how well students have performed a task, typically to determine a final *grade*. Given their different aims, it has been suggested that for clarity the two forms of assessment are best referred to as *formative feedback* and *summative grading* [8].

Formative feedback plays an important role in education. Of all items on the Course Evaluation Questionnaire, the provision of helpful feedback was seen the one factor that clearly distinguished between the best and worst courses [30, 29]. It has also been shown that substantial learning gains can be achieved by innovations designed to strengthen frequent feedback students receive [9, 39]. Interestingly, it has been reported that students pay more careful attention to feedback when there are no associated marks [9], or put another way "marks" reduced student attention to formative feedback.

There are a number of conditions that need to be met for assessment to assist with student learning [18]. The conditions can be grouped into three main points, as shown in the following list:

- Assessment tasks are aligned with intended learning outcomes, and provide students with sufficient work to ensure they engage appropriately with the required learning.
  1. Tasks provide students with enough work to require sufficient time on the task.
  2. Tasks direct students to spend appropriate time and effort on the most important aspects of the unit.
  3. Completing tasks is likely to engage appropriate kinds of learning.
- Feedback is constructive in nature, providing information that students will be able to use to develop their understanding of associated concepts.
  4. Feedback is both timely and sufficiently detailed.
  5. Feedback focuses on demonstrated learning outcomes, and on actions students can control.
  6. Students receive the feedback while it is still relevant, and they are able to incorporate the feedback or seek further assistance.
  7. Feedback needs to be in line with the purpose of the assessment tasks, and relate to its criteria for success.
  8. Feedback should also relate to the student conception of the task, and their understanding of what they are supposed to be doing.
- Students utilise the feedback.
  9. Students must receive, and pay attention to, the feedback.
  10. Feedback should influence students' future actions.

While it has been suggested that one assessment item can play both a formative and summative role [10], students do not view feedback accompanying summative work as formative [33]. Rather these comments were seen as a way of understanding the rationale of the person grading the tasks, and were not used to help students with future tasks.

Additionally, summative and formative assessment are likely to elicit different responses from students [8]. With formative feedback, the ideal strategy for students is to draw attention to their misunderstandings so that they can receive feedback relevant to advancing their current level of understanding. Conversely, this strategy would be detrimental to the student if the task is also performing a summative role. In this case, the summative grading encourages students to hide their own misunderstandings in order to maximise the grade they are awarded. In extreme cases, students plagiarise other's work in an attempt to hide their own misunderstandings, something that does not make sense when the work is formative in nature.

If used effectively, formative feedback can be used to focus students on gaining required levels of understanding. The issue with summative assessment during the teaching period is that marks are typically final, giving the student little incentive to address any feedback they receive alongside the summative grade. With this arrangement, mistakes are punished with lower grades rather than being seen as opportunities to improve learning, a key idea in constructivist learning theories. Where summative assessment can be delayed until after the teaching period, formative feedback can be used to help students develop their understanding of unit concepts. Students can submit work and receive feedback aimed at helping them improve their understanding. These formative tasks help students build toward the final summative assessment, and so students are encouraged to address the feedback they receive and to learn from their mistakes. By engaging with this learning process, students will develop deeper understandings that can help them achieve better results in their final summative assessment.

2.4 Applications of Constructive Alignment

Constructive alignment has been applied in a range of fields, including applications of constructive alignment in the area of computer science and software engineering. In this area constructive alignment has been discussed in relation to teaching concurrency [11] and introductory programming [36, 16]. A brief summary of each is presented below for comparison with the approaches presented in this paper.

Brabrand [11] provided a very detailed discussion of how tasks had been aligned to outcomes for the teaching of concurrency. In this work, programming projects played an important part of the unit, providing students with an opportunity to apply their understanding of concurrency in a practical sense. Assessment consisted of a group project and a multiple choice examination. Results indicated an improvement in the quality of student work in the projects, results, and student satisfaction.

Thota and Whitfield [36], as well as Gaspar and Langevin [16], have reported on the use of constructive alignment in teaching introductory programming. Thota and Whitfield [36] described the development and delivering of an object oriented introductory programming unit that aligned with cognitive and affective learning outcomes. Assessment in the unit used a group project, programming assignments, quizzes, and an examination which were all aligned to the stated unit learning outcomes. While the work of Gaspar and Langevin [16] discussed how the principles of constructive alignment represented a need to explicitly cover programming thought processes in teaching introductory programming, but did not discuss an overall application of these principles to the development of an associated unit of study.

These approaches are all representative of common approaches to constructive alignment, but do all remain primarily teacher-focused and the approaches to assessment do not offer students opportunities to learn from their mistakes, an important feature in more fully realising constructivist learning theories. In these approaches, teaching staff are responsible for setting unit learning outcomes, performing the alignment, and selecting evidence students can use to demonstrate their understanding. Assignments and examinations are completed and submitted, playing a primarily summative role. In these scenarios, students are not required to interact directly with the unit's intended learning outcomes, and so the carefully selected verb may have no impact on the students approach to learning. So while instructive in terms of aligned curriculum, these approaches are limited in how they have embedded constructivist learning theories in their approaches to assessment.

3. CASE STUDY 1: INTRODUCTION TO PROGRAMMING

Introduction to Programming is a large first undergradu-

ate programming unit at Swinburne University of Technology offered to a broad range of students studying engineering, science and game development. Most students undertake the unit in their first semester, with most having no prior programming experience. The objective of the unit is to introduce students to procedural programming concepts, with most students then studying object oriented programming in their second semester.

The following are the main topics covered in this unit:

- Compilers, programs, and source code
- Functions and procedures
- Variables and constants
- Structured programming concepts: sequence, selection, and repetition
- Arrays, records, enumerations and pointers
- Modular and functional decomposition

The introduction to programming unit had the following intended learning outcomes:

1. Apply code reading and debugging techniques to analyze, interpret, and describe the purpose of program code, and locate within this code errors in syntax, logic, style and/or good practice.
2. Describe the principles of structured programming, and relate these to the syntactical elements of the programming language used and the way programs are developed.
3. Construct small programs, using the programming languages covered, that include the use of arrays, functions and procedures, parameter passing with call by value and call by reference, custom data types, and pointers.
4. Use modular and functional decomposition to break problems down functionally, represent the resulting structures diagrammatically, and implement these structures in code as functions and procedures.

3.1 Approach to Unit Delivery and Assessment

Introduction to Programming was redesigned to encompass the principles of constructive alignment in 2008 [13], with iterative improvements being made each teaching period [12]. The unit's method of assessment and its approach to the delivery of teaching and learning material have changed with the aim of improving the focus on "what the student does" and ensuring that this aligns with the unit's stated learning outcomes.

Teaching periods at Swinburne include 12 weeks of classes, followed by a 3-week examination period. Classes in Introduction to Programming included a 2-hour lecture and a 2-hour laboratory session each week across the 12 teaching weeks. Students were strongly encouraged to attend both the lectures and the laboratory classes, though attendance was not mandatory. Lecture and laboratory classes were adapted to help incorporate constructive learning theories. The focus shifted from delivering content to providing students with guidance on programming concepts, and the process of programming.

Lectures typically included a short presentation on that week's programming concepts using a story-based approach [4, 25] followed by an interactive lecture demonstration [31]. Students were actively encouraged to contribute to the coding sessions, providing input into the programs to be created and helping identify and solve issues as they arose. To help support this, additional material was provided via a series of short videos on language syntax and an accompanying textbook that outlined the necessary details students were likely to require in order to succeed at the weekly tasks.

Laboratory classes were used to help guide students through getting started with weekly tasks, and to provide them with feedback on work they had submitted. A range of group and individual tasks were carried out in these sessions, with students being actively encouraged to discuss their understanding with their classmates. Teaching staff discussed their feedback on submitted tasks with students individually, helping ensure each student understood how they were progressing with the unit, and the areas they needed to concentrate their study on.

Both staff and students were involved in the process of aligning student activity with unit learning outcomes. In developing the unit, teaching staff designed weekly tasks to align student activity with the intended learning outcomes. During the delivery, students undertook the tasks and developed evidence that they had achieved the stated learning outcomes. Then, at the end of the teaching period, students reflected on the tasks they had completed and how these tasks enabled them to demonstrate their understanding. In this way, students and staff worked together to enhance the alignment of activities: staff by planning the tasks and students by indicating how those tasks helped them achieve the learning outcomes.

Assessment reinforced the principles of constructive alignment using a portfolio-based approach to assessment [13]. Tasks students completed were submitted each week for teaching staff to provide formative feedback. The feedback students received could be used to improve their work, and develop their understanding. Students were encouraged to incorporate feedback they received and, where necessary, to resubmit the work for additional feedback on the changes made. Through this process, students developed a portfolio of work that demonstrated their capabilities by the end of the teaching period.

Grades were determined by a criterion-referenced assessment of their portfolios [12], with clear criteria being provided for each grade category. To receive a Pass grade, students needed to pass a test and submit a portfolio that demonstrated coverage of all of the intended learning outcomes. Higher grades were achieved by demonstrating deeper understanding, including demonstrating the ability to apply the unit's concepts to the development of a custom program and through conducting a small research project and reporting on their findings.

3.2 Use of Formative Feedback

Formative feedback was central to the system used to teach this unit. Student grades were determined entirely from their perfomance in the portfolio submitted at the conclusion of the teaching period, effectively resulting in removing *marks* during the teaching period. Instead, tasks during the teaching period were designed to aid student learning and enable feedback to be acted upon and incorporated by students.

Figure 1 illustrates the process teaching staff used to deliver the unit. Teaching staff engaged students with the teaching and learning activities and provided tasks designed to aid students in developing their understanding. Students engaged with these tasks and activities, and through this process developed evidence of their learning. Artefacts cre-

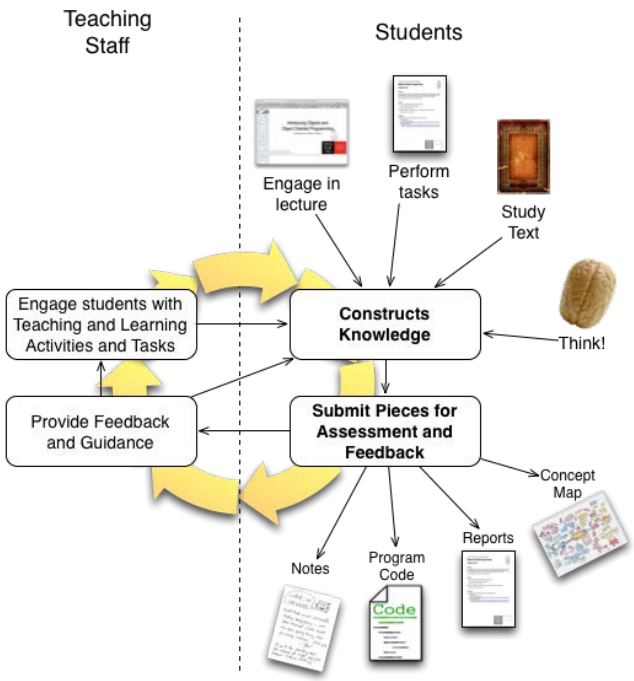

Figure 1: Illustration of delivery process adapted from unit assessment overview documents.

ated by students were submitted to teaching staff for assessment and formative feedback. This enabled staff to guide students efforts, and provide constructive feedback aimed to ensure students were able to develop appropriate knowledge and meet unit learning outcomes.

To help students with their time management, and to assist staff in providing quick responses to student submissions, an online system was developed to help manage the frequent formative feedback cycles present in this approach [40]. The system tracked student progress through the unit's tasks, and provided the means for students to submit work to teaching staff for feedback. Each task submitted by a student could then be signed off as complete by teaching staff when it was demonstrated to an adequate level. Where work submitted demonstrated misconceptions or misunderstandings, staff provided feedback and indicated that the student needed to fix and resubmit the work.

At the end of the teaching period, each student selected work from the tasks they had completed and submitted this as their portfolio for final summative assessment. Students prepare their portfolio by creating a *Learning Summary Report* in which they assess themselves against the unit's assessment criteria, argue that they have met the requirements for a particular grade, and justifying this assessment in reference to the evidence they have provided. Staff then perfomed the final summative grading by assessing the portfolio against the assessment critiera in conjunction with the student's self assessment.

Frequent formative feedback cycles, and the use of the resulting artefacts in student portfolios, meant that staff were able to quickly perform the final summative grading. Each student's portfolio consisted of the work they had been submitting for formative feedback throughout the teaching period. As a result, staff were already familiar with the work, having assessed it in a formative manner earlier in the teaching period. In this way the work students had undertaken could benefit from formative feedback during the teaching period, and then be quickly evaluated after the unit had concluded in to determine the summative grade.

3.3 Results and Student Feedback

Introduction to Programming at Swinburne has been taught using this portfolio-based teaching and learning system for a number of years. During this time, pass rates have improved and student feedback on teaching has remained positive. Results from an associated action research project [12] indicate that student learning appeared to improve over this period as the clarity of the assessment criteria was improved.

Comments related to portfolio assessment in the student feedback on teaching surveys were mostly positive. A number of positive quotes indicated that the portfolio based approach helped to reduce stress associated with other forms of assessment.

> "Overall I was very satisfied with the unit. Having a portfolio based unit and all these weekly tasks reduces the actually stress compared to other units out there. And using games to teach programming was very very effective I personally learnt a lot more in this unit than what I learned in my diploma."

There were also a number of positive comments about the use of the tasks and feedback to guide learning. The following was one example of such a comment.

> "...I really appreciate the task-feedback based teaching method in this unit rather than the dated and extremely ineffective teaching method called examination. The tasks in this unit is well designed and the difficulty is adjusted well for the targeted aim. The P and C tasks are easy enough to learn the basic while the D tasks are just challenging enough but at the same time achievable. Also unlike some other unit the marking in every tasks are flexible enough for students to explore different methods to finish the task. Overall I am very impressed by the organised and effective teaching method."

One student explicitly referred to the formative feedback cycles as being one of the best aspects of the unit.

> "The relaxed nature of all of the teaching staff and how much easier it is to monitor your own progress through the assessments compared to other units is amazing. The whole concept of progressive learning and feedback has made it much easier to focus on achieving the grade that I wish, something a lot harder to do when assessments are a one-shot-only thing. Being able to try and try again to get a program to do what it is meant to do for a passing grade means that the later tasks were considerably easier to tackle because I knew I had effectively mastered the previous knowledge."

A number of students commented that one of the best aspects of the unit was the help provided from the tutors,

indirectly providing positive comments related to the use of formative feedback in the unit.

> "Good tutors, with very helpful one on one feedback. Everything we learned was able to be applied almost immediately and it was easy to build off of the basic content that we learned."

The lack of marks and use of formative feedback also received some negative comments, though a progress tracking system [40] was developed to help mitigate this issue. These comments related to issues with the unit's flexibility, indicating that students are likely to prioritise mark bearing tasks from other units.

> "For me I underestimated the workload, and as there were no specific due dates, I find myself in final week with a vast amount of work to do. Understandably this is my fault, but I assume there are other people with me, who because there is no due dates will put tasks off to work on other units which have constant due submissions."

Workload was also an issue raised by students. The use of *frequent* formative feedback required ongoing work, as distinct from units which use a smaller number of graded assessment tasks.

> "Time management, possibly reduce the amount of tasks to be completed each week to allow first time programmers to have a chance at Distinction and High Distinction"

Other issues included concerns about the clarity of the instructions in the task sheets, and the general difficulty of learning to program. Overall, however, the comments were generally positive with around 65% of students responding with a mean score of 8.61 out of 10 for satisfaction with the unit being higher than the faculty average of 7.54.

4. CASE STUDY 2: INTRODUCTION TO SOFTWARE ARCHITECTURE

Introduction to Software Architecture is a 7.5 ECTS unit that forms the cohort of specialization in software engineering at IT University of Copenhagen. This unit is open to full-time and part-time studies programs offered by the university. There are certain pre-requisites for this unit, and the unit covers all the major topics of an introductory unit on software architecture in many Universities. The objective of the unit is to provide postgraduate students with a sound technical exposure to the concepts, principles, methods, and best practices in software architecture.

The following are the main topics covered in this unit:

- Introduction to the fundamentals of software architecture.
- Software architecture and quality requirements of software systems.
- Fundamental principles and guidelines for software architecture design, architectural styles, patterns, and frameworks.
- Methods, techniques, and tools for describing software architecture and documenting design rationale.
- Software architecture design and evaluation processes.
- Rationale and architectural knowledge management in software architecture.
- Approaches and tools for designing and evaluating software architectures for state-of-the-art technologies such as cloud-computing and service-orientation, and mobile computing.
- Future challenges and emerging trends in software architecture discipline.

The software architecture unit had the following intended learning outcomes (ILO) defined by following the SOLO taxonomy defined for IT University of Copenhagen, Denmark.

1. Argue the importance and role of software architecture in large scale software systems
2. Design and motivate software architecture for large scale software systems
3. Recognise major software architectural styles, design patterns, and frameworks
4. Describe a software architecture using various documentation approaches and architectural description languages
5. Generate architectural alternatives for a problem and select among them
6. Use well-understood paradigms for designing new systems
7. Identify and assess the quality attributes of a system at the architectural level
8. Motivate the architectural concerns and approach for families of products
9. Discuss and evaluate the current trends and technologies such as model-driven, service-oriented, and aspect-oriented architectures
10. Evaluate the coming attractions in software architecture research and practice

4.1 Unit Delivery and Assessment Approach

The unit consists of 14 weeks of teaching consisting of lectures and exercises. The unit also involved guest lectures, student presentations, and project work. The lectures were designed along the lines of seminar style where the students were expected to take active participation. While the attendance in most of the lectures were not mandatory, the students were strongly encouraged to attend the lectures and exercise sessions.

Assessment consisted of a number of mandatory assignments, and a written examination. Student grades, however, were determined entirely on their performance in the 4 hour written examination. This strategy was used due to the requirements of involving an external censor for all graded assessment tasks. This makes it almost impossible to have incremental assessment throughout the unit. The other assessment items are then used as a hurdle requirement, with students needing to perform satisfactorily on designed mandatory learning activities before they can be considered qualified for sitting the examination. Each student must satisfactorily perform on all three components of the mandatory assignments in order to qualify for the final examination. Following were the three types of mandatory assessments used.

- Individual assignments (Two)
- Group presentations (One)
- Group project (Two phases)

Apart from these mandatory learning activities to assess the qualification for sitting in the examination, there were

several learning activities designed as class exercises. Those class exercises were undertaken by students every week in a 2 hour session except on two occasions where all 4 hours were allocated to learning exercises, namely: Software Architecture Design and Software Architecture Evaluation.

The reason for allocating 4 hours was that 2 hour session may not be sufficient for performing these types of activities. All the learning activities (i.e., mandatory and class exercises) were designed to align with the unit's intended learning outcomes and the students' learning and progress was monitored and regular general feedback was provided to students with individual feedback provided on request. The mandatory activities were assessed for their satisfactory completion and on a few occasions, some students were asked to resubmit their mandatory assignments' solutions. The general strengths and weaknesses found in the mandatory assignments submissions were discussed in the exercise classes and students were invited to have open discussion and reflections.

Each of the assessments and learning activities were carefully aligned with the unit learning outcomes. Students gave class presentations in groups, worked on generating scenarios for various software systems, wrote up user personas, and documented software designs.

4.2 Use of Formative Feedback

In the software architecture unit, formative feedback was used to help students prepare sufficiently for the final examination. Students undertook a number of class exercises and assignments aimed to help them engage meaningfully with the unit's learning outcomes. For the class exercises, students were expected to work on one or two learning activities related to the topic that would have been taught a week prior to the class exercise and then they were encouraged to take those learning activities home either to complete or improve. The learning activities given as class exercises were designed to prepare students for carrying out the assignments and/or perform better for the examination. For example, two of the class exercises were generating quality attribute scenarios based on a given set of quality attributes; the second exercise was to design a high level design diagram using deign patterns that can help satisfy the scenarios generated in the first class exercise. The students were provided with general feedback on both exercises in the following weeks and encouraged to seek personal feedback in one-to-one session if required. Both tasks from these class exercises were part of the assignments on designing software architecture for a given system's requirements and there were examination tasks to generate five quality attribute scenarios (worth 5 marks) and design a high level architecture using known design patterns (worth 20 marks out of 100). The examination was an open book examination in which the students were allowed to bring all relevant material from the class exercises, assignments, and lecture notes. Since the examination format and the nature of questions were provided to the students from the beginning of the unit, the students had a very strong motivation to incorporate feedback in the subsequent class exercises and assignments as they were preparing the material that they could use as support during the examination.

4.3 Results and General Feedback

We assert that pass rate in a unit can be one of the significant indicators of students' learning. The Software Architecture unit at the IT University of Copenhagen has been taught using the student-centred approach described in this paper for several years. The pass rates have been very good with a majority of the students getting above average marks (i.e., high pass or credit). Table 1 shows the results for this unit. There have been only 2 or 3 cases of failure each year in this unit. In Denmark relative grading is forbidden by the legal framework and, as a result, there is always an absolute grading. The grading of the final examination is carried out in collaboration with an external censor as per the University's requirement for quality assurance. The main instructor and the external censor grade each students' answering booklet and come up a grade for each of the students. The grades between the two assessors are then compared and where there is a discrepancy of more than 2 or 3 points, or the discrepancy can change the grade classification for a particular student, the instructor and the external censor have a face-to-face meeting to discuss the grade to be awarded. By exchanging their notes and arguments for grades, the final grade is reached via consensus.

The grading scheme defined by the legal framework consists of six classifications of graded: 0, 2, 4, 7, 10, and 12. These grades can be converted into alphabets for which the marks can vary slightly. For example, units may have 45% and below for grade 0 (F), though the norm is usually 50% and below. For the software architecture unit, the grades used were F (up to 45%), E (46%-50%), D (51%-64%), C (65%-75%), B (76%-85%), A (86%-100%). This grading scheme was used in 2012, 2013 and 2014.

As far as learning the key concepts and practical techniques through major assignments and class exercises are concerned, the student appeared to have appreciated the approach used in informal feedback and discussions, though a large majority of students would like to have more time for class exercises. Since this unit runs in the evening, it has not been possible to increase class duration for the exercise classes despite our appreciation that architecture design and evaluation tasks usually take time to complete. However, the teaching team encouraged each student to continue class exercises outside of the allocated class hours and to bring back their work for discussion and feedback the follow week. There was a mix response in terms of availing the opportunity to discuss the exercises, with only a few students seeking further discussion or individual feedback during the allocated time or scheduling a meeting with assigned teaching staff.

One of the key learning activities was designed to provide the students an opportunity to evaluate each other's architecture during a 4 hours evening session called "software architecture evaluation evening". Each team was given the architecture designed by another team for analysing and evaluating before and during the class. These sessions were very popular over the years, with students finding them to be excellent opportunities to learn how to critically analyse and evaluate architectures of others and how to present and defend their own architecture design decisions.

Given the students performance on the class exercises and major assignments, it was clear that the learning activities were well designed for the planned learning outcomes and generally appreciated by the students despite some concerns of workload being on the high end and not having sufficient time for completing tasks during the scheduled classes. How-

Table 1: Student Results from the Software Architecture Unit.

	Grade					
Year	A	B	C	D	E	F
2014	2 (6%)	14 (40%)	11 (31%)	4 (11%)	2 (6%)	2 (6%)
2013	0 (0%)	2 (10%)	8 (38%)	6 (29%)	2 (10%)	3 (14%)
2012	3 (10%)	4 (14%)	12 (41%)	4 (14%)	4 (14%)	2 (7%)

ever, the students' performance on the final examinations were a good indicator that most students were successful in acquiring the expected knowledge, understanding, and skills from the taught topics of software architecture.

5. DISCUSSION

5.1 Making Assessment Formative

Both of the units presented used assessment aimed to be formative in nature; aimed at developing student understanding rather than assessing learning outcomes. The two approaches can be compared using Gibbs conditions related to formative feedback [18]. The assessment approach in the introductory programming unit addressed all of Gibbs conditions necessary for feedback to be formative. Weekly tasks ensured students had sufficient work to engage appropriately with each of the concepts covered (meeting condition 1 to 3). Feedback aimed to be constructive in nature, being provided each week (meeting condition 4), and was aimed at helping students address misunderstandings and assisting them in overcoming obstacles related to completing each task (meeting conditions 5 to 8). Students were then motivated to utilise this feedback, being required to correct issues identified in their solutions before tasks were marked off as complete (meeting condition 9 and 10).

Similarly, the class exercises and assignments in the architecture unit were designed to enable students to engage meaningfully with the unit's learning outcomes (meetings conditions 1 to 3). The general, and one-to-one, feedback was provided after students had completed the class and assignment tasks, with adequate time for them to incorporate feedback before the final examination (meeting condition 4). Feedback was designed to help students correct any misconceptions identified and to assist them in their preparation for the final examination (meeting condition 5 to 8). The open book nature of the exam also provided a means to motivate students to incorporate feedback they received. However, while it was strongly recommended that students act upon the feedback and there was a strong justification for students to act on the feedback to improve their learning and submissions as above mentioned, only a few were required to do so, meaning that conditions 9 and 10 may not have been met for all students.

One notably similarity between the two approaches is the change in role of the formative tasks at the end of the teaching period. With the portfolio based approach the work students complete, in the formative part of the unit, become summative at the end of the unit when they are submitted in the students portfolio. Similarly, with the examination based approach, the artefacts created from the formative tasks could be used by students in their final summative examination. In this way, both approaches encourage students to incorporate feedback, with the portfolio based approach directly using the resulting artefacts in determining final student grades, and the examination based approach indirectly using these due to the open book nature of the final examination.

5.2 Removing Marks

Summative assessment aims to assess the results of teaching and learning, assessing how well students are able to perform a task. Attaching marks to an assessment task means that, from the student's perspective, the task will play a summative role and feedback is not seen as formative [33]. Removing marks can help focus students on the formative nature of the assessment, but runs the risk that students will not engage meaningfully with the task. The two cases presented here offer different means of motivating students to complete tasks without the need to have associated marks; with the portfolio approach utilising these tasks for summative assessment at the end of the unit, and the examination based approach having satisfactory performance on the formative tasks as a hurdle requirement necessary to be eligible to sit the examination.

It appears that often marks are attached to tasks during the teaching period primarily as a means of motivating students to engage meaningfully with these tasks. While the curriculum, from the students perspective, is always defined by the assessment [30], the two approaches presented in this paper indicate that this assessment can better utilise formative feedback during the teaching period by removing marks from these tasks. When tasks do not carry marks, students are better able to focus on the feedback they receive [9], and academics can help guide student learning throughout the teaching period. Delaying the determination of the summative grade to the end of the teaching period gives students the best opportunity to learn from their mistakes, and to develop appropriate understandings by engaging with tasks and incorporating feedback from teaching staff.

Removing marks from assessment also helped create a positive learning environment where students could see that staff and students were on the same side: working together either to prepare for the final examination or to prepare work for the final portfolio assessment. Tasks completed during the teaching period did not attract marks, and so feedback was not seen as justification for why marks were removed [33] but instead could be seen as advice on how to improve understanding or task outcomes. In this way critical criticism could become constructive, with students being encouraged and rewarded for taking feedback on board and using it to improve their work. This made it easier for staff to correct a wide range of issues identified in student submissions, as feedback was not constrained to a marking scheme.

The changed role of the assessment tasks during the teaching period also changed the kinds of discussions staff had with students about the feedback they received. Previously, students would often argue for additional marks where they did not agree with the marks associated by staff. These conversations were rarely productive, with the focus on marks

awarded and the marking scheme, not on the understanding the student had demonstrated. Removing marks also removed these kinds of discussions, helping to focus discussions on how the student can improve, or better demonstrate, their understanding.

5.3 Challenges

Each of the approaches presented has different challenges for staff and students. The primary concern for the portfolio approach related to staff and student workload. Students found it challenging to keep up with the weekly tasks, and could fall further behind when aspects of task submissions needed to be fixed and resubmitted. Similarly, the weekly feedback was a large workload for the teaching staff, with staff having to provide feedback on student progress each week during the teaching period. This weekly feedback was, at least partially, offset by not having any spikes in feedback due to assignments and examinations.

The examination based approach had fewer issues related to workload but placed greater pressure on student's performance in the final examination. Students generally find final exams stressful and prefer a more modular approach to assessment [35]. However, where an exam-based approach is used, open-book exams have been found to have lower levels of student anxiety as compared to closed-book exams and exams where students may bring in a cheat sheet [17].

5.4 Recommendations

Applications of Constructive Alignment should aim to do more than align assessment tasks with unit learning outcomes. student-centred approaches need to be created by incorporating aspects from constructivist learning theories. Accepting the central importance of the student in constructing their own knowledge requires a change from content delivery styles of teaching and learning to approaches that aim to guide and support student activity. In this context, teaching and learning systems that can enable the use of formative feedback cycles are likely to result in positive teaching and learning environments that encourage and reward students for engaging authentically with the associated tasks.

Removing marks from tasks during the teaching period changes the focus of these tasks, helping staff and students to use these in a formative manner. Portfolio based approaches to assessment can then be used to change the role of these tasks from formative to summative at the end of the teaching period. This approach allows students to benefit from the formative nature of the assessment during the teaching period, while also allowing staff to quickly reassess their work summatively at the end of the unit by using knowledge gained throughout the formative feedback cycles. Open book examinations offer an alternative that enables work during the teaching period to be formative but does place greater stress on students for whom the final examination is now the one means of assessing them summatively. However, in both cases the students are better able to learn from the feedback provided as all of the feedback can be directed toward helping enhance understanding, without the need to justify marking decisions.

Where it is not possible to delay summative assessment to the end of the unit, formative feedback could be supported by enabling students to resubmit work after it is graded. This would enable these summative tasks to play a formative role, though this may result in students looking to maximise the marks received in the task without aiming to address underlying issues with their understanding.

6. CONCLUSIONS

Constructive Alignment aims to focus on "what the student does" to encourage and reward students for engaging deeply with teaching and learning activities. These principles work by focusing on innovations in assessment, aligning assessment with unit learning outcomes and building on constructivist learning theories. Most papers on applications of Constructive Alignment focus on the alignment of the curriculum, indicating how assessment tasks relate to unit learning outcomes. While interesting, the alignment of activities addresses only one part of the principles of Constructive Alignment with constructivist learning theories typically receiving little focus.

This paper presents two applications of Constructive Alignment from the areas of Computer Science and Software Engineering. While both units aligned teaching and learning activities with unit learning outcomes, the focus of this paper is on how these units use formative feedback to better enable students to learn from their mistakes: a central theme to constructivist learning theories. Removing marks from assessment tasks during the teaching period helped to ensure these tasks played a formative role: with the formative tasks being incorporated into a portfolio in one case, and being used as preparation for an examination in the other.

Approaches such as those presented in this paper, provide a means for staff to help *guide* student learning. At their heart these approaches involve students undertaking teaching and learning activities, and staff providing advice and guidance (formative feedback) to help direct future student activity. Through this process, staff and students work together to better help students to achieve the unit's learning outcomes. In this process alignment is a requirement, rather than a specific feature, and guidance and support are the tools used to foster student learning.

7. REFERENCES

[1] J. R. Anderson, L. M. Reder, H. A. Simon, K. A. Ericsson, and R. Glaser. Radical constructivism and cognitive psychology. *Brookings papers on education policy*, (1):227–278, 1998.

[2] C. Argyris. Theories of action that inhibit individual learning. *American Psychologist*, 31(9):638:654, 1976.

[3] J. Armarego. *Constructive Alignment in SE education: aligning to what?*, pages 15–37. ACM, 2009.

[4] C. Atkinson. *Beyond Bullets Points: using Microsoft® Office PowerPoint® 2007 to create presentations that inform, motivate, and inspire*. Microsoft Press, 2007.

[5] J. Biggs. Enhancing teaching through constructive alignment. *Higher Education*, 32:347–364, 1996.

[6] J. Biggs and C. Tang. Assessment by portfolio: Constructing learning and designing teaching. *Research and Development in Higher Education*, pages 79–87, 1997.

[7] J. B. Biggs and K. F. Collis. *Evaluating the Quality of Learning: The SOLO taxonomy (Structure of the Observed Learning Outcome)*. Academic Press New York, 1982.

[8] J. B. Biggs and C. Tang. *Teaching for quality learning at university*. McGraw-Hill Education, 4th edition, 2011.

[9] P. Black and D. Wiliam. Assessment and classroom learning. *Assessment in Education*, 5(1):7–74, 1998.

[10] B. S. Bloom. Some theoretical issues relating to educational evaluation. *Educational Evaluation: New Roles, New Means (National Society for the Study of Evaluation Yearbook, Part II)*, 68:26–50, 1969.

[11] C. Brabrand. Constructive alignment for teaching model-based design for concurrency. *Transactions on Petri Nets and Other Models of Concurrency I*, pages 1–18, 2008.

[12] A. Cain. Developing assessment criteria for portfolio assessed introductory programming. In *Proceedings of the 2nd IEEE International Conference on Teaching, Assessment and Learning for Engineering*, pages 55–60. IEEE, 2013.

[13] A. Cain and C. J. Woodward. Toward constructive alignment with portfolio assessment for introductory programming. In *Proceedings of the first IEEE International Conference on Teaching, Assessment and Learning for Engineering*, pages 345–350. IEEE, 2012.

[14] S. A. Cohen. Instructional alignment: Searching for a magic bullet. *Educational Researcher*, 16(8):16–20, 1987.

[15] A. Q. F. Council. *Australian qualifications framework*. Australian Qualifications Framework Council, South Australia, second edition, 1 2013.

[16] A. Gaspar and S. Langevin. An experience report on improving constructive alignment in an introduction to programming. *Journal of Computing Sciences in Colleges*, 28(2):132–140, 12 2012.

[17] A. Gharib, W. Phillips, and N. Mathew. Cheat sheet or open-book? a comparison of the effects of exam types on performance, retention, and anxiety. *Psychology Research*, 2(8):469–478, 8 2012.

[18] G. Gibbs and C. Simpson. Conditions under which assessment supports students learning. *Learning and Teaching in Higher Education*, 1(1):3–31, 2004.

[19] M. Haigh. Writing successfully for the journal of geography in higher education. *Journal of Geography in Higher Education*, 37(1):117–135, 6 2013.

[20] D. H. Jonassen. Objectivism versus constructivism: Do we need a new philosophical paradigm? *Educational technology research and development*, 39(3):5–14, 1991.

[21] E. Martin, M. Prosser, K. Trigwell, P. Ramsden, and J. Benjamin. What university teachers teach and how they teach it. *Instructional Science*, 28:387–412, 2000.

[22] F. Marton and R. Säljö. On qualitative differences in learning- II outcome as a function of the learner's conception of the task. *British Journal of Educational Psychology*, 46(2):115–127, 6 1976.

[23] F. Marton and R. Säljö. On qualitative differences in learning: I - outcome and process. *British Journal of Educational Psychology*, 46(1):4–11, 2 1976.

[24] F. Marton and R. Säljö. Approaches to learning. In F. Marton, D. Hounsell, and N. Entwistle, editors, *The Experience of Learning: Implications for teaching and studying in higher education*, pages 39–58. Edinburgh: University of Edinburgh, Centre for Teaching, Learning and Assessment, 3rd internet edition, 2005.

[25] R. E. Mayer. *The Cambridge Handbook of Multimedia Learning*. Cambridge University Press, 2005.

[26] D. C. Phillips. The good, the bad, and the ugly: The many faces of constructivism. *Educational researcher*, 24(7):5–12, 1995.

[27] R. Phillips. Challenging the primacy of lectures: The dissonance between theory and practice in university teaching. *Journal of University Teaching & Learning Practice*, 2(1), 2005.

[28] J. Piaget. *Psychology of Intelligence*. Routledge & Kegan Paul, London, 1950.

[29] P. Ramsden. A performance indicator of teaching quality in higher education: The course experience questionnaire. *Studies in Higher Education*, 16(2):129–150, 1991.

[30] P. Ramsden. *Learning to Teach in Higher Education*. Psychology Press, 1992.

[31] M. J. Rubin. The effectiveness of live-coding to teach introductory programming. In *Proceeding of the 44th ACM technical symposium on Computer science education*, SIGCSE '13, pages 651–656, New York, NY, USA, 2013. ACM.

[32] M. Scriven. The methodology of evaluation. In R. W. Tyler, R. M. Gagne, and M. Scriven, editors, *Perspectives of Curriculum Evaluation*, volume 1, pages 39–83. Rand McNally, Chicago, IL, 1967.

[33] K. Skinner. Bridging gaps and jumping through hoops: First-year history students expectations and perceptions of assessment and feedback in a research-intensive uk university. *Arts and Humanities in Higher Education*, 13(4):359–376, 10 2014.

[34] L. P. Steffe and J. E. Gale. *Constructivism in education*. Lawrence Erlbaum Associates, NJ, 1995.

[35] I. Suto, G. Elliott, N. Rushton, and S. Mehta. Course struggle, exam stress, or a fear of the unknown? a study of a level students?' assessment preferences and the reasons behind them. *Educational Futures*, 6(2), 2014.

[36] N. Thota and R. Whitfield. Holistic approach to learning and teaching introductory object-oriented programming. *Computer Science Education*, 20(2):103–127, 2010.

[37] R. W. Tyler. *Basic Principles of Curriculum and Instruction*. University of Chicago Press, 1969.

[38] C. Vrasidas. Constructivism versus objectivism: Implications for interaction, course design, and evaluation in distance education. *International Journal of Educational Telecommunications*, 6(4):339–362, 2000.

[39] D. Wiliam. Formative assessment: Getting the focus right. *Educational Assessment*, 11(3-4):283âĂŞ289, 2006.

[40] C. J. Woodward, A. Cain, S. Pace, and F. K. J. Jones, A. Helping students track learning progress using burn down charts. In *Proceedings of the 2nd IEEE International Conference on Teaching, Assessment and Learning for Engineering*, pages 104–109. IEEE, 2013.