Constructive Alignment for Teaching Model-Based Design for Concurrency

(a case-study on implementing alignment in Computer Science)

Claus Brabrand

IT University of Copenhagen,
Rued Langgaards Vej 7, DK-2300 Copenhagen S, Denmark
brabrand@itu.dk
http://www.itu.dk/people/brabrand/

Abstract. *"How can we make sure our students learn what we want them to?"* is the number one question in teaching. This paper is intended to provide the reader with: i) a general answer to this question based on The Theory of *Constructive Alignment* by John Biggs; ii) relevant insights for bringing this answer from theory to practice; and iii) specific insights and experiences from using constructive alignment in teaching model-based design for concurrency (as a case study in implementing alignment).

Key words: Teaching, Student learning, Constructive Alignment, The SOLO Taxonomy, Model-Based Design for Concurrency.

1 Introduction

This paper is intended to show *how* The Theory of Constructive Alignment [2] provides a compelling answer to the number one question in teaching:

"How can we make sure our students learn what we want them to?" (Q1)

Specifically, to illustrate *how* the theory can be used in the context of teaching model-based design for concurrency, to guide and maximize student learning; and, to provide *incentive* and *support* for student learning in a direction intentionally chosen by a teacher.

The paper is divided into two parts. Part 1 briefly gives a general introduction to The Theory of Constructive Alignment and The SOLO Taxonomy [3]. The essence of this part is also available as a 19-minute award-winning short-film by the author, entitled *"Teaching Teaching & Understanding Understanding"* [5]. Part 2 is the main part and shows how to apply the theory to a specific case; namely, to teach a 5 ECTS[1], seven week, undergraduate course on model-based design for concurrency at the University of Aarhus, Denmark. The course has been taught four times by the author using FSP [10] for modeling, Java for

[1] *European Credit Transfer and Accumulation System* (one academic year is 60 ECTS)

implementation, and the book [10] for introducing relevant concepts, problems, and solutions. The course ran twice before the implementation of alignment (in 2004 & 2005), and twice after (in 2006 & 2007); each year with a group of 20–30 students. The paper concludes by giving a comparison of teaching the course "pre-" vs. "post-alignment".

2 Part 1: The Theory of Constructive Alignment

The Theory of Constructive Alignment [2] provides a compelling answer to `(Q1)`. The theory is developed by John Biggs and has its roots in *curriculum theory* and *constructivism* [11]; the idea that the learner's *actions* define what is learned and that knowledge is actively constructed by the individual through interaction with the external world (see [13] and [8]). It is a *systemic theory* that regards the total teaching context as a whole, as a *system*, wherein all contributing factors and stakeholders reside. To understand the system, we need to identify and understand the parts of the system and how they interact and affect one another. The Theory of Constructive Alignment provides just that for the teaching system; it provides relevant and prototypical models of the parts that ultimately enable us to *predict* how the teaching system reacts when we change various aspects of our teaching. It is also a theory of motivation and of planning that looks at teaching far beyond what goes on in the classroom and auditorium.

However, before we present constructive alignment as "the solution" to `(Q1)`, we need to look closer at (models of) the main parts of the system; the students, the teachers, and of cognitive processes.

As with all models (just like the models we use in concurrency) they might seem a bit simplistic or crude at first. Nonetheless, they are highly instructive for us to get an idea of what the system looks like and what causes and effects we may be up against as a teacher.

2.1 Student Models

In his book, *"Teaching for Quality Learning at University"* [2], John Biggs has identified and personified two prototypical student models classified according to their motivation for being at university, immortalized as *"Susan and Robert"*:

Susan is *intrinsically* motivated. She likes to get to the bottom of things and often reflects on possibilities, implications, applications, and consequences of what she is learning. She uses high-level learning activities such as *reflecting*, *analyzing*, and *comparing* that continually deepen her understanding.

Robert, on the other hand, is *extrinsically* motivated. He is not interested in learning and understanding in itself; he just wants to pass exams, so that he can get a degree, so he can get a (decent) job. To this end, he will cut any corner, including sticking with lower-level learning activities, such as *identifying*, *note-taking*, and *memorizing* as long as they suffice.

It is important to note that a given student may embody any combination of these two prototypes and that it may vary according to the area of interest. For this reason it is often advantageous to think of them as *strategies* (as in *"The Susan Strategy"*), rather than actual *persons*.

As a teacher, it is not Susan we need to watch out for. Faced with a curriculum, she basically teaches herself; in fact, we almost cannot prevent her from learning. Rather, it is Robert we need to pay attention to; in particular, to the learning activities he is employing (before, during, and after teaching).

Our challenge as a teacher is to engage Robert and get him to use higher-level learning activities; i.e., make him *behave* more like Susan. The good news is that it is actually possible to do something about Robert (or rather, Robert's learning). We shall shortly explain how to, systemically speaking, positively change the system so as to (have him) change his *behavior*. But before we do that, we need to look at the situation from a teacher perspective.

2.2 Teacher Models

John Biggs also has a few prototypical models of the teachers; this time three (increasingly desirable) models of teachers according to their main focus in teaching, known as the *"three levels of thinking about teaching"* [2]:

The level 1 teacher is concerned with what students *are*. He operates with a binary perspective; a student is either (inherently) good xor bad. The exam is a diagnostic means to "sort the good students from the bad" after teaching. This perspective is essentially deferring the responsibility for lack of learning; in particular, the teacher can no longer do anything about it: "it's just the way students are; either good or bad" (i.e., independent of the teaching).

The level 2 teacher is concerned with what the teacher *does*. A teacher at the second level is preoccupied with acquiring an armory of techniques, "tips'n'tricks" along with visual and technological aides, in order to enhance *his* performance. While this perspective is a dramatic improvement to the first, it is still independent of *student learning* which is incorporated directly in the third and final level.

The level 3 teacher is concerned with what a student *does* (before, during, and after teaching). He is adopting a *student-learning focus* and will judge all pedagogic dispositions according to how they affect student learning.

Again, a given teacher may embody combinations of these characteristics.

2.3 Learning Models

In 1949, one of the most influential American educators, Ralph W. Tyler said:

> *"Learning takes place through the active behavior of the student: it is what he does that he learns, not what the teacher does."*

The idea that knowledge is *transmitted* from an (active) teacher to a (passive) learner is dead. There is increasing evidence that what is learned is intimately tied to which actions are performed by the learner and that knowledge is *actively constructed* (see [13] and [8]). In fact, John Biggs defines good teaching [2] directly as a function of student activity:

- Quantitative -		- Qualitative -	
SOLO 2 "*uni-structural*": - paraphrase - define - identify - count - name - recite - follow (simple) instructions - ...	SOLO 3 "*multi-structural*": - combine - classify - structure - describe - enumerate - list - do algorithm - apply method - ...	SOLO 4 "*relational*": - analyze - compare - contrast - integrate - relate - explain causes - apply theory (to its domain) - ...	SOLO 5 "*ext'd abstract*": - theorize - generalize - hypothesize - predict - judge - reflect - transfer theory (to new domain) - ...

Fig. 1. Sample competence verbs from *"The SOLO Taxonomy"*. Improvements in learning outcomes occur *quantitatively* at SOLO 2–3 and *qualitatively* at levels 4–5.

> *"**Good teaching** is getting most students to use the higher cognitive level processes that the more academic students use spontaneously."*

Teaching is about activating students; getting them to use *higher cognitive level processes*. For this we need a model of understanding, cognition, and quality of learning. There are many such models; e.g., *"The SOLO Taxonomy"* [3], *"The BLOOM Taxonomy"* [4][2], and Klopfer's models of student behavior [9]. However, I have chosen to present only one of these models, namely The SOLO Taxonomy, since it has been deliberately constructed for research-based university teaching and converge on research at its fifth and highest level.

The SOLO taxonomy (*"Structure of the Observed Learning Outcome"* [3]), distinguishes five levels of cognitive processes according to the cognitive processes required to obtain them. Although there is a close relationship between the levels of the SOLO taxonomy and the levels in Jean Piaget's (hypothetical) cognitive structures, the former was designed to evaluate learning outcomes and cognitive processes, the latter for describing the developmental stages of individuals, especially children (see [3] and [12]). The five levels are (in increasing order of complexity, each level prerequisitionally building upon the previous):

SOLO 1 (aka. "the pre-structural level") At the first level, the student has no understanding, uses irrelevant information, and/or misses the point altogether. Although scattered pieces of information may have been acquired, they

[2] Note: *"The BLOOM Taxonomy"* was originally designed to guide representative *selection* of items on a test, rather than *evaluating* the quality of learning outcomes.

will be unorganized, unstructured, and essentially void of real content or relation to a relevant issue or problem.

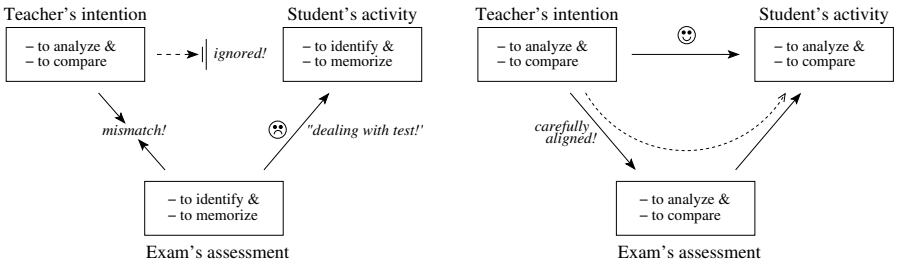
SOLO 2 (aka. "the uni-structural level") At the second level, a student can deal with one single aspect. A student may make obvious connections and hence have the competence to *recite*, *identify*, *define*, *follow* simple instructions, and so on.

SOLO 3 (aka. "the multi-structural level") A student at level three can now deal with several aspects, but they are considered independently. A student may have the competence to *enumerate*, *describe*, *classify*, *combine*, *structure*, *execute* procedures, and so on.

SOLO 4 (aka. "the relational level") At the relational level, a student may now understand relations between several aspects and understand how they may fit together to form a whole. A student may thus have the competence to *compare*, *relate*, *analyze*, *apply*, *explain* things in terms of causes and effects, and so on.

SOLO 5 (aka. "the extended abstract level") At the fifth and highest level, a student may generalize structure beyond what was given, essentially producing new knowledge. A student may perceive structure from many different perspectives, transfer ideas to new areas, and may have the competence to *generalize*, *hypothesize*, *theorize*, and so on.

Figure 1 shows a *non-exhaustive* list of common verbs from the SOLO taxonomy.



(a) An *unaligned* course. (b) An *aligned* course.

Fig. 2. An unaligned vs. aligned course.

2.4 Constructive Alignment

We now have the ingredients and models to understand the system and why constructive alignment is a compelling answer to `(Q1)`.

Definition 2 (constructive alignment): A course is said to be *constructively aligned* [2] when:

- the *intended learning outcomes* (ILOs) are stated clearly;
- the ILOs are explicitly communicated to the students;
- the exam assessment(s) match the ILOs; *and*
- the teaching form(s) match the ILOs.

The solution is to constructively align courses (the name "alignment" comes from the fact that the following elements are all pointing in the same direction):

$$\textit{exam assessment} \;\approx\; \textit{intended learning outcomes} \;\approx\; \textit{teaching form}$$

To appreciate this solution, let us first have a look at the problems with an *unaligned course* where there is a mismatch between the intended learning outcomes and the exam assessment. After this, we will see how constructive alignment remedies this situation.

Unaligned course Figure 2(a) illustrates an example of an *unaligned* course. Here, it is the teacher's intention that the students learn how to *analyze* and *compare*. However, the nature of the exam used is such that it measures something else; in this case, the ability to *identify* and *memorize*. The problem with this arrangement is that Robert will soon realize the minimal requirements, totally ignore the teacher's intended learning outcomes, and only study for what is directly required of him on the exam. This *"backwash effect"* is appropriately referred to as Robert *"dealing with the test"*.

Aligned course Figure 2(b) depicts an *aligned* version of the course. Here, the teacher has carefully aligned the exam with the intended learning outcomes such that it assesses precisely those (in this case, the ability to *analyze* and to *compare*). We get a commuting diagram; Robert's goal of passing the course will invariably lead him past learning the intended objectives. This way, we are effectively using Robert's (extrinsic) motivation to pass courses, to make him learn.

Now Robert is *motivated* to learn, but he still needs the support. This is where the form of teaching comes in; the other aspect of constructive alignment is to also align the *teaching form* with the intended learning outcomes and exam. During a course, students would ideally "train towards the exam". The challenge then becomes choosing—perhaps several different—adequate forms of teaching in which the students best practice the skills and competences intended and measured.

In a constructively aligned course Robert now has the *support* (from the teaching form) and *incentive* (from the exam assessment) to learn like Susan.

This was a brief and general introduction to The Theory of Constructive Alignment. In the following, we will have a look at *how* these ideas can be applied to improve the teaching of a Computer Science course on Model-Based Design for Concurrency.

3 Part 2: Constructive Alignment for Teaching Concurrency

Adhering to the principle of the Chinese Proverb:

> *"Give a man a fish and he will eat for a day. Teach a man to fish and he will eat for a lifetime."*

this section will not attempt to present "*a* perfectly aligned course", but rather, to illustrate *how* the principle of constructive alignment can be useful and used for judging the relevance of and selecting different forms of assessment and teaching.

3.1 From Content to Competence

First though, I want to motivate and advocate a shift from thinking courses in terms of *content* to thinking in terms of *competence*.

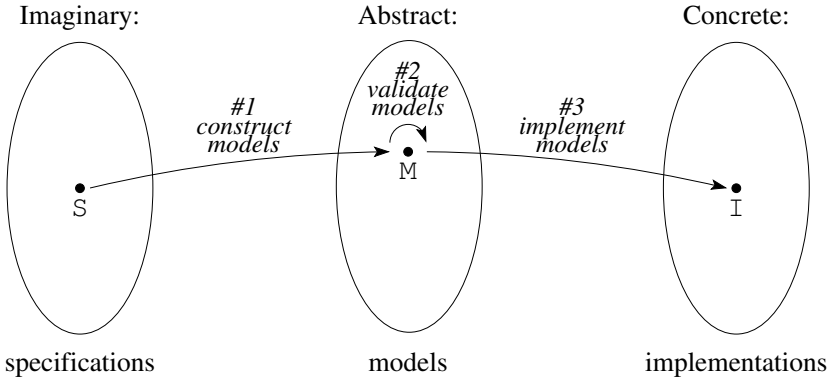
Content Traditionally, many courses specify the aims of a course as a *content description*; listing course-specific concepts that are "to be understood". This was also the case in earlier versions of my concurrency course (before exposure to the theory of alignment, that is). It essentially stated that the goal of the course was for the students to *understand* a bunch of concurrency concepts such as "interference" and "deadlock" (see Figure 3(a) for the exact formulation).

The problem with general "understanding goals" via content descriptions is that teachers and students may not (in fact, usually do not) have the same *interpretation* of the intended learning outcomes. Teachers and examiners—being products of a research-based teaching tradition—will immediately agree; that what is really meant by "understanding deadlock" is, for instance, the competence to *analyze* programs for deadlock, *explain* possible causes and effects, and *predict* consequences of possible solutions. However, this is tacit knowledge. A student—unfamiliar with the traditions—is likely to interpret the same content description at an entirely different level; e.g., as the competence to *recite* conditions for deadlock and *name* standard solutions. However, even if students and teachers did agree on an interpretation, we already know from the theory of alignment, that Robert's learning activity will still ultimately be dominated by the constitutional effect of the exam (cf. Figure 2(a)).

Competence is inherently operational and captured by verbs as opposed to content by nouns. Competence is knowledge plus the capacity to act upon it; to *use* attained understanding of a topic to inform behavior and act accordingly. The SOLO levels provide a taxonomy of appropriate verbs for describing intended learning outcomes as a hierarchy of competences. Thus, in our ultimate intended learning outcomes, we are not aiming for (passive) knowledge of content, but (active) competence.

Aim:
The purpose is to give the students a thorough knowledge of models, systems, and concepts in concurrency (cf. contents below), such that this may be used in the realization of solid solutions to realistic and practical problems.
Contents:
Processes, threads, interaction, interference, synchronization, monitors, deadlock, safety and liveness properties, forms of communication, and software architecture for systems and concurrency.

(a) Pre-alignment: course aims (given as a content description).



(b) Course philosophy: the model-based design process.

STEP no.	COMPETENCE: *After the course, students are expected to be able to...:*	SOLO level
n/a	• ~~***memorize*** content~~;	~~2~~
#1	• ***construct*** models from specifications;	3
	• ***apply*** standard solutions to common concurrency problems;	4
	• ***relate*** models and specifications;	4
#2	• ***test*** models w.r.t. behavior (using tool support);	2
	• ***define*** relevant safety/liveness properties for models;	2
	• ***verify*** models w.r.t. safety/liveness properties (using tools);	3
	• ***analyze*** models (and programs) w.r.t. behavior;	4
	• ***compare*** models (and programs) w.r.t. behavior;	4
#3	• ***implement*** models in familiar programming languages; *and*	3
	• ***relate*** models and implementations.	4

(c) Post-alignment: intended learning outcomes (based on the SOLO taxonomy)

Fig. 3. Pre- and post-alignment course description.

3.2 Course Philosophy: Model-Based Design

There is obviously a wide spectrum of perspectives on concurrency and thus on possible concurrency courses; ranging from the study of abstract categorical frameworks for concurrency process calculi to semaphore protocol programming. However, as hinted in the title of this paper, the overall philosophy and activity in the course investigated in this paper is centered around models, using a *model-based design* approach.

Before presenting the intended learning outcomes as competences based on The SOLO Taxonomy, I want to spend a few lines on motivation. In general, as a teacher I need to provide students with a solid answer for what they get out of following the course; what it is they will be able to *do* after the course, why that is important, and what advantages those competences will give them. Also, I need to spend time communicating this answer to the students. If I cannot "sell the course" to the students who have not actively elected the course, they will be less motivated to spend time on it. Thus, when teaching model-based design for concurrency, I need to provide my students with a solid answer to the (very appropriate) question:

"Why bother learning about model-based design for concurrency!?"

Here is a short summary of the motivational answer I give my students early on in the course, motivating a "model-based design approach" to concurrent software development. Concurrent programming is much more difficult than sequential programming; systems are inherently non-deterministic and parallel; the concurrency is conceptually harder to grasp and adds—along with complexity—a whole new range of potential errors such as interference, deadlock, starvation, un-intended execution traces, unfairness, and priority inversion. In the presence of all these errors, models come to the rescue. Models offer a means for offline reasoning through a formal modeling language to read, write, and talk about models (to gain understanding of a system), run-time testing (to gain confidence), and automatic[3] compile-time property verification (to gain safety).

The model-based design process, as depicted in Figure 3(b), advocates that systems are better built by first constructing models from specifications (step #1), then validating the models constructed (step #2), and only then implementing those validated models as concrete systems (step #3). The quality of the final resulting system constructed is, of course, tied to the "appropriateness" of the intermediary steps through the models.

3.3 Intended Learning Outcomes

With the overall philosophy of the course in place, I need to *operationalize* it and express it in terms of concrete evaluable competences. Here, one needs to carefully avoid the temptation to use so-called *understanding goals* (e.g., such as *"to understand X"*, *"be familiar with Y"*, or *"have a notion of Z"*), for the

[3] *"Never send a human to do a machine's job"*, A. Smith (The Matrix, 1999).

simple reason that we cannot measure them. General understanding goals should be turned into measurable competence. Note that understanding is, of course, a requisite for competence.

Figure 3(c) presents the intended learning outcomes expressed as competences based on The SOLO Taxonomy and directed towards the students. The description starts with the formulation:

"After the course, students are expected to be able to ..."

Note how this formulation places the learning focus on the students and that it is directly expressed in terms of *competence* (i.e., "to be able to..."). This line is then followed by the individual competences to be learned during the course. The first, "to memorize content" (which is at SOLO level 2) is explicitly included as a non-goal to send a clear message to the students that this competence will not help them during the course and exams.

This is followed by the ten actual intended learning outcomes for the course listed along with their corresponding SOLO level. The competences are divided into the three steps related to model-based design process (#1 to #3). Note how each competence is expressed using an active *verb* (highlighted in boldface) and a passive noun/noun-phrase, expressing: "what is it the students are expected to be able to *do* (verb) with *content* (noun)". The application of standard solutions to common concurrency problems covers issues such as semaphores, mutual exclusion, synchronization, deadlock, and the reader/writer protocol.

I will not go further into the particular choice of intended learning outcomes here, because this is not the point of this paper. Rather, the point is to show how such intended learning outcomes can be used to provide *incentive* and *support* for student learning in a direction intentionally chosen by a teacher, as explained in the following.

3.4 On Aligning the Assessment (with the ILOs)

When I learned about constructive alignment in 2005, the exam of my concurrency course consisted of a group project during the last two weeks of the course, and an individual multiple-choice test at the end of the course, each counting 50% towards the final grade. This happened to coincide with my preferences if I were to choose freely among all reasonable forms of evaluation for measuring the intended learning outcomes of Figure 3(c), as reported in the following along with my experiences.

On Aligning the Project In the pre-alignment courses, it was also my intention that emphasis be placed on the model-implementation relationship which had also been clearly communicated to the students. However, since the aims of the course were given by a traditional *content description* (Figure 3(a)), this was not reflected in explicit intended learning outcomes, nor was it listed as explicit criteria for project grading. In the end, I received some projects with no apparent relation between the two. It was as if the construction of the model

and implementation had been approached independently and pursued in two altogether different directions, defying the whole purpose of model-based design.

In the 2006 course, I tried to apply the idea of constructive alignment. I formulated explicit intended learning outcomes around which the exam was carefully centered and on which the teaching was based. To *relate* was explicitly included as an intended learning outcome and explicitly included on the exam. The product was a project entitled "The Banana Republic" which was a synthesis-oriented project where the students had to construct a system via the model-based design paradigm. Figure 7 and Figure 8 show the specification, task, report requirements, and evaluation criteria of the 2006 project. The project was explicitly *designed* to evaluate all the competences of Figure 3(c), except the more analytical competences; i.e., *to analyze* and *compare* models and programs. In my opinion, such competences are more appropriately evaluated in a multiple-choice test as explained below. The projects received in 2006 generally had a better correspondence between model and implementation.

On Aligning the Multiple-Choice Test The two analytical competences not directly addressed in the project (*to analyze* and *to compare* models and programs), are more appropriately evaluated in a multiple-choice test. The main advantage is that in a multiple-choice test one is free to prefabricate (even contrived) models whose main purpose is to exhibit more interesting and challenging aspects and behaviors than the students are likely to come upon during the model-based construction process. Since I believe these two competences are important, and not guaranteed to be required in the project, I have to explicitly examine the students in them. Thus, I have devoted an independent test solely to them.

I used the Multiple Choice Tool (MCT [7]) to automatically permute questions and choices, to evaluate the answers, and to ensure that the grading was statistically robust and based on provably sound principles.

Earlier tests asked seemingly innocent questions such as the one found in Figure 4(a). Although this seems like a perfectly reasonable question to ask and for which the students should know the answer, it has dramatic implications on learning. The problem is that it is possible to get by with *memorization*. Hence, Robert is free to "deal with the test" and direct his study effort towards memorizing content (recall Figure 2(a)). Note that information about the sufficiency of surface understanding may also be rumored by former students exposed to similar questions in earlier courses.

After the introduction of alignment, later tests were carefully centered around the competence *to analyze* and *to compare* models. For examples of such questions, see Figures 4(b) and 4(c). (Please note that we do not expect the reader to understand the details of the FSP models; for details of the FSP modelling language, we refer to [10].) It should be obvious that these are high-level questions for which lower-level activities such as memorization no longer suffice. By construction, they depend on the capacity to analyze and compare models. Some questions were also testing the ability to analyze and compare (Java) programs.

I still use the the memorization question. However, now it instead serves as a "non-goal"; as an example of a type of question *not* appearing on the final multiple-choice exam, hence the strikeout in Figure 3(c).

3.5 On Aligning the Teaching Form (with the ILOs)

In constructively aligning my form of teaching, I use a combination of five different teaching activities. Specifically, I use:

(1) ***lectures*** to introduce the students to fundamental concepts and to show applications of standard solutions to common concurrency problems in terms of models and implementations (based on [10]);
(2) ***modeling and programming lab*** as a means for students to gain hands-on practical experience in *constructing*, *implementing*, *testing*, and *verifying* models, *defining* properties, and *applying* standard solutions to common concurrency problems (here a TA is present and acts as a consultant);
(3) ***theoretical exercise classes*** as a means for the students to learn how to *apply* variations of common solutions to standard problems (here the students get feedback from a TA who supervises and facilitates the class);
(4) ***weekly hand-ins*** in the form of small compulsory exercises wherein the students are asked to *construct* and *implement* models with special emphasis on *relating* models and implementations (here the students train for the project and receive individual feedback on their hand-ins from a TA); *and*
(5) ***multiple-choice sample questions*** as a means for the students to learn to *analyze* and *compare* models (here the students train for the multiple-choice exam. The questions are given without the correct answers, to maximize student activation.

Note how the real training of competences (i.e., practicing of verbs) takes place, not during the lectures, but in the four other *student-centric* learning activities. This disposition is consistent with the ideas of constructivism; that knowledge is (actively) constructed by the students themselves according to their behavior. There is a big difference between a student (passively) listening to a lecture on application and the student performing the applying himself. During the lectures, I try my best to engage and activate the students using various techniques such as one-minute papers [1], two-minute neighbor discussions, and a three-minute student structural recapitulation at the end to encourage active participation. However, such level 2 "tips'n'tricks" are beyond the scope of this paper.

Structurally, the course iterates through the model-based design process many times with the introduction of each new concurrency concept (a structure also taken in [10]). The advantage of doing it this way, rather than a division according the steps model-based design process (i.e., #1, #2 and #3), is that the students get to practise the overall process many times over and incorporate insights and feedback from previous the iterations. The project is thus essentially the last, unsupervised, iteration.

In earlier versions of the course, teaching activities (4) and (5) above were missing, along with the training in and feedback on those competences. Also,

What are FSP programs compiled into by the LTSA tool?:

- a ☐ Stateless Machines.
- b ☐ Finite State Models.
- c ☐ Infinite State Models.

(a) Assesses competence: *"to **memorize** content"* (i.e., bad alignment).

Given the following FSP model `M`, *safety property* `S`, and *liveness property*, `L`:

```
RESOURCE = (get -> put -> RESOURCE).

P = (printer.get -> (scanner.get -> copy -> printer.put -> scanner.put -> P
                    |timeout -> printer.put -> P)).

Q = (scanner.get -> (printer.get -> copy -> printer.put -> scanner.put -> Q
                    |timeout -> scanner.put -> Q)).

||M = (p:P || q:Q || {p,q}::printer:RESOURCE || {p,q}::scanner:RESOURCE).

property S = (p.printer.get -> p.printer.put -> S
             |q.printer.get -> q.printer.put -> S).

progress L = {p.copy, q.copy}
```

Which of the following property relationships are satisfied?:

- a ☐ `M` $\models$ `S` and `M` $\models$ `L` *(i.e., M satisfies both S and L)*
- b ☐ `M` $\models$ `S` and `M` $\not\models$ `L` *(i.e., M satisfies S, but not L)*
- c ☐ `M` $\not\models$ `S` and `M` $\models$ `L` *(i.e., M satisfies L, but not S)*
- d ☐ `M` $\not\models$ `S` and `M` $\not\models$ `L` *(i.e., M satisfies neither S, nor L)*

(b) Assesses competence: *"to **analyze** models w.r.t. behavior"* (i.e., good alignment).

Let two FSP processes, `CRIT` and `LOCK`, be given:

```
CRIT = (acq->crit->rel->CRIT).
LOCK = (acq->rel->LOCK).
```

Now, consider the two different systems, `SYS1` and `SYS2`, defined below:

```
||SYS1 = ({x,y}:CRIT || {x,y}::LOCK).
||SYS2 = ({x,y}::CRIT || {x,y}:LOCK).
```

Which of the following traces is *invalid* for `SYS1` and *valid* for `SYS2`?:

- a ☐ The empty trace (containing no actions).
- b ☐ `x.acq`
- c ☐ `x.acq ; x.rel`
- d ☐ `x.acq ; x.crit ; x.rel`
- e ☐ `x.acq ; y.crit ; x.rel`

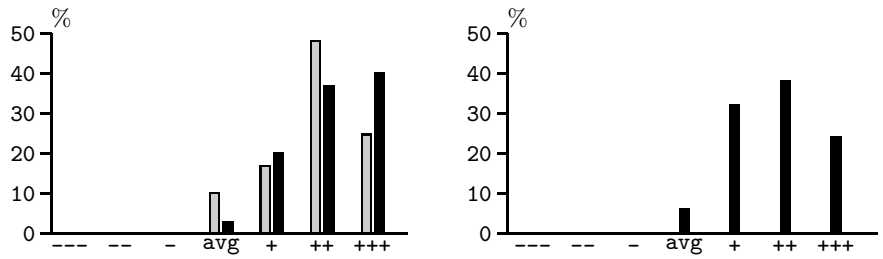
(c) Assesses competence: *"to **compare** models w.r.t. behavior"* (i.e., good alignment).

Fig. 4. Unaligned and aligned sample multiple-choice questions (each question always has exactly one correct answer).

the lectures (1) were more one-way communication and did not explicitly incorporate student activation. Finally, the activities (1), (2), and (3) were never deliberately structured around intended learning outcomes (since these were never consciously established).

4 Conclusion

In the following, I attempt to compare "pre-" vs. "post-alignment" courses and report my experiences divided into subjective and objective measures. However, a few reservations should be kept in mind before attempting to reason about the causes and effects of alignment: there are many factors involved that may vary from year to year; as all teachers I gain more experience over time, the student population varies, and the "Susan/Robert ratio" may vary from year to year.



(a) Student satisfaction. (b) Student proficiency.

Fig. 5. Self-reported student satisfaction and confidence (on a 7-step scale): pre-alignment in gray (Concurrency 2004 & 2005); post-alignment in black (2006 & 2007). Pre-alignment data is not available for student proficiency.

Subjectively, it is my experience that the theory of constructive alignment provides a solid and constructive answer for `(Q1)`. It provides insights on where and how to optimize the teaching system for student learning in making sure the students have the necessary *incentive* and *support* for learning. It is also my own personal experience that the course and the quality of the projects handed in by the students improved significantly with alignment. Also, before alignment, I primarily acted on my intuition, whereas alignment has influenced my behavior and I am now making conscious and informed choices. I am now aware of different pedagogical possibilities and, perhaps more importantly, of the implications different dispositions are likely to have on student learning.

Objectively, I have quantitative data from student self-evaluation questionnaires, reporting on *student satisfaction* (with the teaching) both before and after the implementation of alignment:

	Year	#students	#evaluations	Percent
Pre-alignment	2004	29	10	63%
	2005	26	24	92%
Post-alignment	2006	17	16	94%
	2007	22	19	90%

In 2004, the evaluations were conducted on the last day which featured a "bonus lecture" on concurrency abstractions in `C++` which was not on the exam curriculum; hence the low attendence and number of evaluations.

Figure 5(a) plots student satisfaction as reported by themselves on a 7-step scale in a questionnaire at the end of the course; the gray bars depict the distribution of the answers in the pre-alignment courses (2004 and 2005), while the black bars illustrate the situation for the post-alignment courses (2006 and 2007). The students appear slightly more satisfied after alignment which can also be taken to mean that implementing alignment did not compromise student satisfaction.

However, student satisfaction should not be over-estimated; although a positive sign, it need not correlate with student learning. It is much more interesting to compare student self-reported proficiency in the area of study after the course. Unfortunately, I did not evaluate student proficiency before I got introduced to educational theories, notably to *evaluation theory* [6]. Hence, only the post-alignment (black) data is available as presented in Figure 5(b). Although generally positive, without the pre-alignment data it is hard to draw firm conclusions as to the effect of alignment from the evaluations.

Step number	Competence (abbreviated):	pre-alignment (SOLO levels)	post-alignment (SOLO levels)
n/a	• ~~***memorize*** content..~~	2	-
#1	• ***construct*** models..	3	3
	• ***apply*** solutions..	4	4
	• ***relate*** model/spec..	-	4
#2	• ***test*** models..	2	2
	• ***define*** properties..	2	2
	• ***verify*** models..	3	3
	• ***analyze*** models..	-	4
	• ***compare*** models..	-	4
#3	• ***implement*** models..	3	3
	• ***relate*** model/impl..	-	4

Fig. 6. Pre- vs. post-alignment courses compared w.r.t. the SOLO levels directly involved. For each objective; when explicitly *tested* and *trained* for, the SOLO level of the objective is indicated (otherwise, a dash "-" is given).

If we compare the pre- and post-alignment courses with respect to the SOLO levels involved (those tested for on the exam and trained for during teaching activities), we get an interesting picture. The two rightmost columns of Figure 6 show the SOLO levels of the learning activities involved in the pre-alignment and post-alignment courses, respectively; a dash "-" is given when the learning objectives were not *tested* and *trained* for. Evidently, alignment has facilitated

a significant increase in the SOLO levels involved, in tune with Biggs' definition of "good teaching". The pre-alignment courses predominantly involved lower-level SOLO 2 and 3 activities with most of the level 4 activities completely missing. The post-alignment courses, on the other hand, managed to explicitly incorporate the intended higher-level-4 objectives (*relate*, *analyze*, *compare*, and *relate*), while discouraging the low-level memorization activity.
In 2006, one of the students wrote the following in the anonymous course evaluation which pretty much captures exactly what I was aiming (and hoping) for:

> **Overall:** *"This course has been awesome! It took me a while to be able to think in models, but I saw the light along the way."*
>
> **Teaching:** *"Lectures have been great, the theoretical exercise classes have been rewarding and the feedback has been immense and insightful"*
>
> **Exercises:** *"I did not have a lot of time to do the exercises, but they seemed relevant from week to week."*
>
> **Project:** *"The mini project was a good and solid exercise in analyzing a problem, making a model and implementing it. A very good exercise!"*

Finally, I believe we need to move away from considering the exam a "necessary evil" to instead recognize and perceive it as a powerful pedagogical and motivational instrument.

Acknowledgments The author would like to acknowledge John Biggs, Bettina Dahl Søndergaard, Torben K. Jensen, Anne Mette Mørcke, Mogens Nielsen, and Michael Schwartzbach for valuable comments and suggestions. Also, thanks to Jacob Andersen, Søren Besenbacher, and Martin Mosegaard, for serving as TAs on the course, providing me with continual feedback.

References

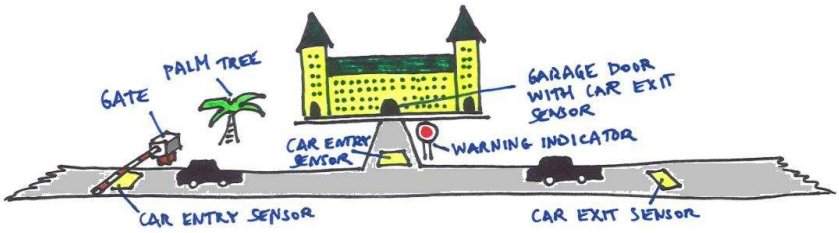
1. Thomas A. Angelo and Patricia K. Cross. *Classroom Assessment Techniques: A Handbook for College Teachers (Jossey Bass Higher and Adult Education Series)*. Jossey-Bass, February 1993.
2. John Biggs. *Teaching for Quality Learning at University*. The Society for Research into Higher Education and Open University Press, 2003.
3. John Biggs and Kevin F. Collis. *Evaluating the Quality of Learning: The SOLO Taxonomy*. New York: Academic Press, 1982.
4. Benjamin S. Bloom. *Taxonomy of educational objectives: the classification of educational goals*. David McKay Company, Inc., New York, 1970. Handbook 1: Cognitive Domain.
5. Claus Brabrand. Teaching Teaching & Understanding Understanding. 19-minute (award-winning) short-film. Available on DVD through Aarhus University Press and online: `[ http://www.daimi.au.dk/~brabrand/short-film/ ]`, October 2006. Written & Directed by Claus Brabrand. Produced by Claus Brabrand and Jacob Andersen.

6. Peter Dahler-Larsen. *Den Rituelle Reflektion - om evaluering i organisationer*. Syddansk Universitetsforlag, 1998.
7. Gudmund S. Frandsen and Michael I. Schwartzbach. A singular choice for multiple choice. In *ITiCSE-WGR '06: Working group reports on ITiCSE on Innovation and technology in computer science education*, pages 34–38, New York, NY, USA, 2006. ACM Press.
8. Jerry Gale and Leslie P. Steffe. *Constructivism in Education*. Lawrence Erlbaum Associates, 1995.
9. Leopold E. Klopfer. Student behavior and science content categories and subcategories for a science program. *University of Pittsburgh, Learning Research and Development Center*, pages 1–62, 1970. WP-54.
10. Jeff Magee and Jeff Kramer. *Concurrency: state models & Java programs*. John Wiley & Sons, Inc., New York, NY, USA, 1999.
11. Ference Marton and Shirley Booth. *Learning and Awareness*. Lawrence Erlbaum Associates, 1997. Mahwah, NJ.
12. John Pegg and David Tall. The fundamental cycle of concept construction underlying various theoretical frameworks. *Zentralblatt für Didaktik der Mathematik (ZDM)*, 37(6):468–475, 2005.
13. Jean Piaget. *Genetic Epistemology*. New York, Columbia University, 1970.

A Project Specification

Here is the specification of the *"Banana Republic"* project, as given to the students:

Banana Republic:



Textual specification:

A one-way road passes by the presidential palace in the *"Banana Republic"*. In order not to delay his excellency, El Presidente, and to make him avoid too close contact with the population, a gate has been mounted (to the west) so that access to the road may be restricted (by closing the gate). Underneath the gate is a car entry sensor which detects cars passing by the gate when it is open. The road also has a car exit sensor (to the east) which detects when cars exit the area in front of the palace. The garage door of the palace is equipped with a sensor to detect when the presidential car is leaving the palace; an entry sensor detects when it enters the main road, and a warning signal (`on`/`off`) indicates whether or not cars are on the road (i.e., whether or not it is safe for the president to enter the road).

You may assume that `N=4` cars drive on the main road and that they "reappear" to the west when they drive away to the east (as in the old PacMan games). Cars may overtake each other, even in the crossing area (which has a capacity of, say, `M=3` cars). You may also assume that his excellency, El Presidente, only leaves the palace and that his car reappears at the palace when he has driven off (to the east).

Your job is to make sure (using a *controller*) that no other cars are on the road in the area in front of the palace at the same time as El Presidente's. The controller receives input from the sensors and may control the gate (`open`/`close`) and warning indicator signal (`on`/`off`).

When El Presidente is nowhere in sight, the gate should be open so the cars may pass into the restricted road without delay, however when El Presidente is coming, he should be allowed to safely enter the road as soon as possible - even in congested rush-hour traffic.

(a) Specification.


```
CAR_ENTRY_SENSOR =
  (car_enter -> CAR_ENTRY_SENSOR).
CAR_EXIT_SENSOR =
  (car_exit -> CAR_EXIT_SENSOR).
```

```
GATE = OPEN,
OPEN = (close_gate -> CLOSED
       |pass_gate -> OPEN),
CLOSED = (open_gate -> OPEN).
```

(b) Processes given. (For details of the FSP modelling language, we refer to [10].)

Fig. 7. Project specification.

Your task: [specification ↦ (unsafe) model ↦ (safe) model ↦ (safe) implementation]:

(a) Construct a model of the (unsafe) `BANANA_REPUBLIC` (i.e., without a *controller*).
(b) Test your model to see that collisions with El Presidente can occur (give trace).
(c) Define a safety property, `NO_CRASH`, that can check that collisions with El Presidente can occur.
(d) Verify that collisions with El Presidente can occur (using the above safety property).
(e) Now construct a *controller* and add it to the system to model a `SAFE_BANANA_REPUBLIC` (such that collisions with El Presidente can no longer occur).
(f) Then verify formally that collisions with El Presidente can no longer occur (with the controller constraining the behavior).
(g) Subsequently add a liveness property, `LIVE_PRESIDENTE`, formally verifying that El Presidente is always eventually permitted to enter the restricted road even in congested rush-hour traffic.
(h) Finally, implement your (safe) model in Java as closely to your model as possible (and give a UML diagram of its structure).

(a) Project task.

Document everything in a small written report which should (at least) include:

(1) Discussions of relevant problematic issues;
(2) Explanations of your solutions and motivations for your solutions;
(3) For step (a), give an explanation of the meaning of all actions in terms of all processes;
(4) For step (a) & (e), a discussion of the relationship between your model and the specification.
(5) For step (h), a discussion of the relationship between your model and your implementation.

The report should be self-contained in the sense that we should be able to understand your solution and the motivations for your solution without having to look into the model or implementation. This means that it should for instance include all necessary and relevant parts of the model and implementation, underlining relevant discussions in the report.

Be concise and to the point (not necessarily "the more explanation the better"); include only issues relevant to the problem at hand (irrelevant issues may subtract points). This is what wins you points (there are no points for an unmotivated solution "out of the blue").

(b) Project report.

The grading is done relative to the *course objectives*; i.e., that you demonstrate the ability to:

- ***construct*** (unsafe and safe) models of the "Banana Republic" (from the specification);
- ***apply*** standard solutions to common concurrency problems in the "Banana Republic";
- ***relate*** your (unsafe and safe) models of the "Banana Republic" to the specification;
- ***test*** your unsafe model and exhibit a collision trace (using the LTSA tool);
- ***define*** the `NO_CRASH` and `LIVE_PRESIDENTE` properties relevant for the "Banana Republic";
- ***verify*** your (unsafe and safe) models wrt. the above properties (using the LTSA tool);
- ***implement*** your safe model in Java; *and*
- ***relate*** your implementation to your safe model.

(c) Project evaluation criteria.

Fig. 8. Project task, report, and grading.