Continuous Activity - A Pedagogical Pattern for Active Learning

Christian Köppe

Hogeschool Utrecht, Institute for Information & Communication Technology,
Postbus 182, 3500 AD Utrecht, Netherlands
christian.koppe@hu.nl
http://www.hu.nl

Abstract. Student assignments often last for a longer period and are conceived so that the students can finish them in this period. However, this requires that the students continuously work on them, but experience shows that this is not always the case. The pedagogical pattern Continuous Activity helps in ensuring that the students indeed are working on the assignment during the whole period and can finish them on time and with sufficient quality.

Keywords: Educational Patterns, Pedagogical Patterns

Introduction

The Pedagogical Patterns Project [1] already published a few pattern languages which cover different aspects of teaching. One language focuses on active learning[7], whereby Active Student is the high-level entry pattern. Using the taxonomy of instructional methods introduced by Baumgartner [5], this pattern can be placed at the module layer of educational actions (layer D). All the other patterns in the Active Learning language can be placed at the lower levels of educational actions: educational ensembles (layer C) and educational scenarios (layer B). Therefore this language mainly addresses problems of how to improve active learning of students regarding shorter periods of time — ranging from minutes up to a few hours. But one aspect is not sufficiently covered yet: to ensure that students stay active over longer periods. We classify this aspect as being a module level problem. The pattern introduced in this work shows a proven solution to this problem and therefore closes a gap in the pattern language of Active Learning.

The pattern uses a version of the Alexandrian pattern format, as described in [3]. The first part of the pattern is a short description of the context, followed by three diamonds. In the second part, the problem (in bold), the background, and the forces are described, followed by another three diamonds. The third part offers the solution (again in bold), consequences of the pattern application — which are part of the resulting context — and a discussion of possible implementations. In the final part of the pattern, shown in *italics*, we present some known applications.

CONTINUOUS ACTIVITY

Fig. 1. A german *Kuckucksuhr* - the bird sings every quarter-hour

You want to have ACTIVE STUDENTS and use assignments which last for a longer period.

❖ ❖ ❖

If students get an assignment and a deadline, they mostly start too late to work on the assignment. They often are not able to finish the assignment in the best possible quality and on time.

Often students have to work on assignments which last for a longer period, ranging from a few weeks to a few months. The scope and workload of these assignments are conceived so that students can finish them in this period with the assumption that they work continuously on them. But most often students believe that they can defer the work until closer to the assignment delivery date. Eventually, when they begin, they do not have enough time to finish it on schedule. This leads to overworking in the last days or even hours before the deadline. Figure 2 shows a typical distribution of student activity. The author presented this Figure at the Dutch Computer Science Education Conference (NIOC) and all attendees described similar perceptions [9].

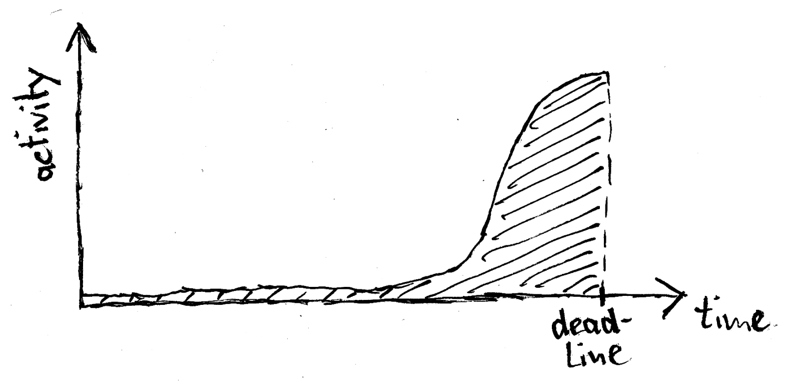

Fig. 2. Typical activity-distribution from students

Temporal Motivation. An explanation for this behaviour can be found in Temporal Motivational Theory (TMT), which is established by Steel and König in [12] and includes aspects of *Picoeconomics* (or *Hyperbolic Discounting*), originally introduced by Ainslie and Haslam. It suggests that people have to choose from different activities which possibly offer rewards. These rewards are undervalued if the events occur in the future, so activities are chosen which will generate immediate rewards, even if these are of lesser value than the future events. But then time comes closer, and people recognize the real value of the once-future events, but are then not able to realistically achieve the goals anymore [12]. They found that their theory applies to individuals, but also to groups.

Groups Work. Larger student assignments are often group assignments (see also the pedagogical pattern GROUPS WORK for this). These assignments are subject to group dynamics. Parts of the assignment are regularly distributed among the students, so every student feels more responsible for her own part and not the whole assignment. Rewards of value are rarely or not given to the other students for their work, so again the motivation to work continuously on the assignment is low. Another aspect is that many group assignments require continuous integration or an integration of all individual work done by the students at the end. If there is not enough time left to do this, then a big-bang integration occurs, leading to an inconsistent and sometimes ill-functioning final product.

Early Phase of Study. Beginning students are particularly prone to this habit, as they often lack the discipline and intrinsic motivation required to overcome this behaviour.

Misleading Experiences. Sometimes students were subject to this behaviour in earlier and smaller assignments, and their experience showed them that even when starting too late working on an assignment that they still were able to finish it with an acceptable grade. This encourages them to repeat this behaviour in later assignments as well, even though these require a much higher amount of work. The students have problems to estimate the amount of time needed for differently-sized assignments.

Unknown Workload. Related to the misleading experiences is the difficulty students have to realistically estimate the workload that is needed to realize a bigger assignment.

❖ ❖ ❖

Therefore: include regular delivery moments of appropriate artifacts to motivate and engage the students to be active over the whole time of the assignment. These artifacts should be of value for the students.

This solution is essentially the application of the technique *Goal Setting* in combination with the earlier described *Temporal Motivational Theory (TMT).* Steel and König state in [12] that "by subdividing a large project into smaller goals, the sum of the parts can be greater than the whole". Defining valuable subgoals for shorter periods leads to an increased motivation per subgoal — which correlates with increased activity per subgoal — and, in the end, to a bigger amount of activity. A higher motivation, stimulated by the grading of these artifacts or other ways of being of value for students, forms the basis of the desired activity-distribution as shown in Figure 3.

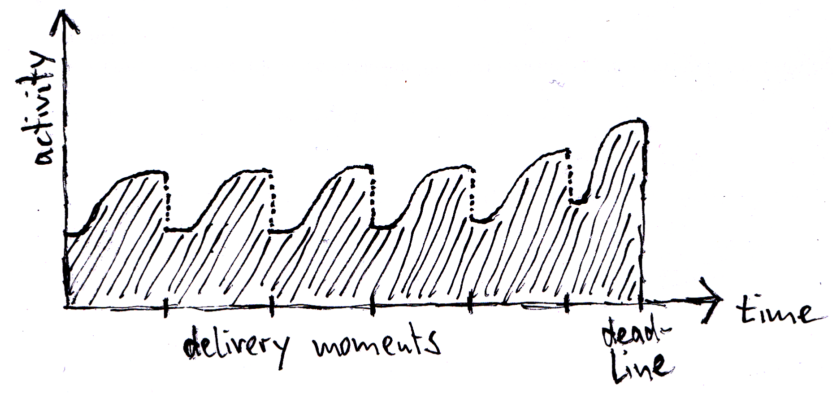

Fig. 3. Activity-distribution with delivery moments

Brewster and Fager also suggest to "break large tasks into a series of smaller tasks" [8], which also has the effect of not overwhelming and discouraging the students on longer-lasting assignments. This way the students are able to realize more parts of their assignments. The value they receive from reaching the sub-goals can be of different kind, where *feedback* (using e.g. formative assessments) and *grades* (using e.g. summative assessments) are the most usual and important ones and often used in combination. We therefore will take a closer look at these two and the kinds of projects where CONTINUOUS ACTIVITY is applicable.

Feedback is of value for the students, as using this feedback will help them to improve their solutions or rework parts of their assignments. At all moments where parts of the assignment are graded, also feedback should be given. Giving feedback is also possible in different ways. Feedback patterns [6] like e.g. PEER FEEDBACK can be used for implementing these different ways. Important is the formative character of the feedback. Barron et al. describe this in [4, pg15] as one of the principles of course design: "...the provision of frequent opportunities for formative assessment by both students and teachers". The students can improve their work during the assignment. The feedback at the first delivery moments can be used to realize EARLY WARNING. Because there are many delivery moments, these can be easily used to EMBRACE CORRECTION.

Grades form another value for the students. Experience shows that it is good practice to weight the first grades less than later ones, especially if the project is the first one for the students which makes use of this pattern. If the students are not used to continuously work on their assignment, they often perform less good in the beginning which leads to a bad grade. If this grade is of a high weight, then the motivation of the students rapidly decreases, which was experienced by us and also colleagues from other universities. A bad first grade which counts just for a small percentage in the total grade is experienced by the students as a warning shot which they still can "iron out".

Brewster and Fager remark that extrinsic rewards — like grades — should only be used sparingly and should be closely related to the accomplished task [8]. It is also important that the rewards should only be given if they are clearly deserved and that they are based on the task and not on comparison to other students.

A variation of this pattern is the application without explicit grading. This variation can be applied when the student approaches the final phase of her study. In most cases an explicit grading of the artifacts is not needed anymore to increase the motivation of the student. Schümmer and Schmolitkzy describe patterns for supervising thesis projects [11]. As a thesis is also a long-running project, it is necessary to ensure a continuous activity of the student writing the thesis. Their pattern — addressing this continuous activity — is PROJECT HEARTBEAT, which makes use of the same solution as presented in this pattern, but does not use the grading part. Instead of defining subtasks or varying deliverables, they monitor the (incremental) progress of the students' thesis project.

Barron et al. use an approach without explicit grading which is similar to this pattern [4] (see section Known Applications).

As all sub-assignments form one big assignment, they also should be seen as a part of a whole. Warren is using holistic assessments to realize this as described in [13]. The holistic approach requires that all assessment criteria are explicit and transparent to students — which is anyway important for student motivation [8] and basis for a FAIR GRADING. This way the students know what is expected of them and it helps the teacher in giving fast feedback. However, using this approach requires that for all sub-assignments explicit assessment criteria *can* be defined. It is for example easier to realize with small software projects which make use of a defined software process (as e.g. OpenUP or Scrum), as these mostly naturally include iteration milestones and process artifacts and for both of these typically transparent and explicit criteria already are — or can be — defined. On the other hand, e.g. writing an essay does not easily provide a couple of defined artifacts or moments which could be used "out of the box". In this case the application of this pattern requires more preparation in advance: the definition of appropriate moments and artifacts, their relations with the overall learning objectives, but also the way these artifacts are to be assessed.

Another consequence of applying this pattern is that it offers the teacher the possibility to react immediately to problems students have with an assignment. Through seeing that specific learning objectives are not achieved by the students in early phases of a project, the teacher can adjust parts of the assignment or the whole assignment. If for example the programming level of the students is not sufficient for a specific assignment, this should become obvious in one of the first delivery moments and the teacher can adjust the programming level of the assignment. Giving extensive feedback takes time, so a good balance has to be found between giving as much feedback as possible and the time required to do so.

But there are also consequences for the students. Through keeping the students continuously active, their need to overwork decreases, especially during the end phase of the assignment. This can actually be seen as the implementation of the eXtreme Programming practice *Sustainable Pace.*

It is important to mention that this pattern should especially applied in the early phases of a curriculum, otherwise the chance of misleading student experiences grows and it becomes more difficult to motivate the students to overcome the behavior described in the problem statement.

This pattern can be used for teaching different domains. Examples are small research projects that could be split up into a research plan, an outline of the project, a literature study, an experiment part, and the final report. Another example is a quantitative statistics project which could be split up in defining the background of the study, the definition of variables, preparing the questionnaire, executing the survey, preparing and validating the results and finally the presentation of the results. So, as long as there are moments of value, like the singing from the bird of a german *Kuckucksuhr* every 15 minutes, this pattern can be applied.

An application of this pattern in Software engineering assignments often include a couple of artifacts which are related to different activities and part of a software process. The Open Unified Process (OpenUP [2]) knows for example process artifacts — the work products in OpenUP terminology — like a use case model, an architecture notebook or implemented physical parts of a system. These artifacts can be graded, using previously defined and communicated criteria. Depending on the focus of the project (the learning objectives), the distribution of the gradings for the total grade has to reflect this focus and can differ per project. A project with the focus on requirements engineering will likely include more iterations for gathering the requirements as well as activities like user workshops, prototyping, or requirements reviews. Here artifacts as well as activities can be used for giving grades. A project with the focus on software architecture will probably include artifacts like an architecture notebook (including architectural artifacts like components, layering, tiers etc. and also a rationale for their justification), key functional requirements, or non-functional requirements. It also will make use of activities like iteration management, architectural prototyping, or design. Again, all these can be used for grading.

It is hard to think of bigger assignments where this pattern can not be applied, as most assignments implicitly offer the possibility to split them up into smaller tasks or evolving project parts (like different versions of a specific architecture model or a thesis). However, these parts should also be related to the learning objectives. If most of these objectives are related to the development of the students' personal skills and can only be graded at the end of an assignment (like reflecting on ones commitment in a team over the whole project or leading a project team), then it is not helpful to apply this pattern, as the teacher has to take the whole project (or assignment) into account.

Another important aspect is that mostly courses are not given in isolation, but often in parallel. As all these courses require activities from the students, the distribution of delivery moments of one of these courses should take the delivery moments of the other courses into account as well. Otherwise there is still a chance of activity peaks which could lead to students focussing on only a few courses and the unwanted activity distribution as described earlier on the other courses.

The implementation of this pattern requires a couple of steps, which are described in the following list.

1. Identify main learning objectives of the assignment and the main timeline (if not already done).
2. Identify the subtasks. Alternatively the students could be asked to identify the subtasks themselves, as they learn more this way. The identification of subtasks could also be determined by constant and regular periods of time.
3. Identify the deliverables per subtask. These can also be new improved versions of earlier deliverables.
4. Define assessment criteria for these deliverables and relate these criteria to the learning objectives. This step also includes deciding on whether using feedback, grading, both of them, or another way of giving a reward to the students.
5. Define the timeline for the subtasks.
6. Communicate the subtasks, deliverables, assessment criteria, and the timeline to the students.
7. While the assignment is running: Assess each subtask deliverable as soon as possible after delivery. Make sure that the assessment results are communicated to the students according to the earlier published assessment criteria. Use these results also always for giving feedback.
8. After the assignment: Evaluate the subtask division and, if necessary, adjust it to improve the students' results.

The author and his colleagues used this pattern for different courses and projects of the software engineering curriculum at the Hogeschool Utrecht. In one project the students were required to use the Open Unified process (OpenUP) for building an auction website. They had to plan short iterations with well defined deliverables as well as delivery moments at the end of each phase (using the milestones of OpenUP). Most of these deliverables (like the architecture notebook) were used for grading while others were only used to give feedback (like iteration plans) so that the students could improve their work. All delivery moments were required. However, in a first run the students did focus only on the graded deliverables. So in a second run of this project all deliverables were taken into account for grading. The end results were — compared to earlier projects without continuous delivery moments incl. grading — of better quality including a good integration of all parts, sufficient testing etc.

Another way to implement this pattern can be the Scrum model for managing student projects, as presented in [10]. Here Scrum is used to force the students to set and achieve short term goals. This correlates with the increase of work amount the students were able to realize while still having the same amount of time [10].

Barron et al. describe in [4, pg284] some experiences with problem- and project-based learning. Their approach is similar to the one described in this pattern, namely using "explicit cycles of assessment, feedback, and revision centered around student-generated products.". The only difference is that they are focusing on giving formative feedback and not using grades.

The author asked the participants of the presentation on pedagogical patterns at the Dutch Computer Science Education conference 2011 (see [9]) if they recognize the problem as stated in the pattern problem statement. All participants agreed on that this is a common problem. The author then asked for the solutions the participants applied. These all included in-between delivery- or contact moments, where the students got something of value for them (either grades, tips, or other feedback). After presenting the sketch in Figure 3, most participants agreed that they use the same idea.

Acknowledgements

I first want to thank my shepherd Yishay Mor for his helpful and eye-opening comments on the first drafts. I also want to thank the participants of the writers workshop at the EuroPLoP 2011: Christian Kohls, Peter Baumgartner, Reinhard Bauer, Andreas Rüping, Claudia Iacob, and Dirk Schnelle-Walka. I'm especially thankful to the first three workshop members because of the excellent discussions we had about educational patterns. The picture of the german *Kuckucksuhr* is courtesy of www.eichwaelder.de.

References

1. Pedagogical patterns project. `http://www.pedagogicalpatterns.org/`, 2010.
2. OpenUP. `http://epf.eclipse.org/wikis/openup/`, 2011.
3. Christopher Alexander, Sara Ishikawa, and Murray Silverstein. *A Pattern Language: Towns, Buildings, Construction (Center for Environmental Structure Series).* Oxford University Press, later prin edition, August 1977.
4. Brigid J S Barron, Daniel L Schwartz, Nancy J Vye, Allison Moore, Anthony Petrosino, Linda Zech, John D Bransford, The Cognition, and Technology Group at Vanderbilt. Doing with Understanding: Lessons from Research on Problem- and Project-Based Learning. *The Journal of the Learning Sciences*, 7(3/4):271–311, 1998.
5. Peter Baumgartner. *Taxonomie von Unterrichtsmethoden: Ein Plädoyer für didaktische Vielfalt.* Waxmann Verlag, Münster, 2011.
6. Joseph Bergin, Jutta Eckstein, Mary Lynn Manns, and Helen Sharp. Feedback Patterns. `http://www.pedagogicalpatterns.org/`, 2011.
7. Joseph Bergin, Jutta Eckstein, Mary Lynn Manns, and Helen Sharp. Patterns for Active Learning. `http://www.pedagogicalpatterns.org/`, 2011.
8. Cori Brewster, Jennifer Fager, and Northwest Regional Educational Laboratory. *Increasing student engagement and motivation: From time-on-task to homework.* Northwest Regional Educational Laboratory, 2000.
9. Christian Köppe. Een tijd-(en grenze) loze manier van onderwijs: Pedagogical Patterns. In *Proceedings of the NIOC 2011 conference*, Heerlen, Netherlands, 2011.
10. Dean Sanders. Using Scrum to manage student projects. *J. Comput. Small Coll.*, 23(1):79, 2007.
11. Axel Schmolitzky and Till Schümmer. Patterns for Supervising Thesis Projects. In *Proceedings of EuroPLoP 2008*, 2008.

12. Piers Steel and Cornelius J König. Integrating theories of motivation. *Academy of Management Review*, 31(4):889–913, 2006.
13. Ian Warren. Teaching patterns and software design. In *Proceedings of the 7th Australasian conference on Computing education - Volume 42*, ACE '05, pages 39–49, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.