Work-in-Progress: An agile approach to formative assessment in higher education

1st Aleksander Larsen Skrede
Department of ICT and Natural Sciences
Norwegian University of Science and Technology
Ålesund, Norway
0000-0002-0534-1010

2nd Øystein Bjelland
Department of ICT and Natural Sciences
Norwegian University of Science and Technology
Ålesund, Norway
0000-0002-2581-9843

3rd Evelyn Honoré-Livermore
Department of Electronic Systems
Norwegian University of Science and Technology
Trondheim, Norway
0000-0002-5664-330X

***Abstract*—Formative assessment is part of a learning process in which the teacher continuously monitors students' learning and provides feedback intended to let students themselves reflect upon what they have learned and where they need to focus more attention. The underlying goal of formative assessment is to lift the students to a higher metacognitive level where they can monitor and guide their learning while simultaneously improving their learning outcomes. However, many courses are planned before the semester starts and may not include methods to continuously improve the teaching. With continuous monitoring and assessment, the philosophy behind formative assessment has some similarities to the agile methodology often employed in software- and product development. This paper presents an agile framework for formative assessment in engineering courses.**

By taking the viewpoint of "Knowledge as a product", this focus ensures: (1) that the students increase their knowledge through sprints and demonstrate it often, and (2), that the methods of teaching and learning are continuously improved through feedback from the students so that the teacher can adapt and improve to enable learning, all supported by formative assessment. Additionally, the agile approach to formative assessment has the potential to improve the non-technical professional skills of students.

***Index Terms*—formative assessment, feedback, agile learning**

I. Background

Formative assessment focuses on providing feedback to learners and teachers throughout the learning process, in contrast to summative assessment, where assessment methods could hinder learning as the focus is sometimes shifted towards attaining good grades instead of useful knowledge [1]. We have adopted the definition of formative assessment provided by Black and Wiliam [2, p. 9]:

> *Practice in a classroom is formative to the extent that evidence about student achievement is elicited, interpreted, and used by teachers, learners, or their peers, to make decisions about the next steps in instruction that are likely to be better, or better founded, than the decisions they would have taken in the absence of the evidence that was elicited.*

In addition to provide an iterative approach to learning, Clark [3] argues that formative assessment also reinforces students' abilities to self-regulate their learning. However, a recent study by Schildkamp et al. [4] discusses that formative assessment in the classroom has shown mixed effects regardless of its evident potential. The authors suggest that the chosen strategy for providing formative assessment is critical, and the lack of positive effects could stem from improper or partial implementation in the classroom. They argue that formative assessment should not be an addition to traditional teaching but a complete restructuring in how teaching is performed.

Moreover, formative assessment requires feedback that is well timed and effective. Hattie and Timperley [5] argue that feedback in education should answer three major questions, namely: *Where am I going? How am I going? and Where to next?* They also distinguish between feedback about the task (FT), processing of the task (FP), self-regulation (FR) and about self as a person (FS), and argue that FS is the least effective, FR and FP enhance deep learning, while while FT is effective when task information is sufficient for improving strategy processing.

Another aspect, pointed out by Gagatsis, is that students may have a compartmentalized view of the various courses and topics they learn in their studies [6]. Compartmentalized views may be prevalent in multidisciplinary programs, such as where hardware meets software, design meets manufacturing, or mathematics meets applied sciences.

Studies performed in Australia suggest that employers are unsatisfied with recent graduates' capabilities concerning non-technical professional skills [7]. An emphasis was placed on effective communication, time management, open-mindedness, and the ability to learn from errors and receive feedback. Employers rated these skills as essential but stated they lacked in graduates. While these skills are not typically found in the curriculum of engineering programmes, the universities could promote development of these skills by changing the way teaching and learning is done in the classroom [8].

This paper describes an agile approach to teaching where formative assessment is incorporated while simultaneously attempting to break down the various mental compartments to give the students a more holistic view of the topics. A bachelor's programme in automation will be used as an exam-

ple of how the framework can be applied. An emphasis should be placed on the curriculum's applicability and support for the students to develop practical skills. Additionally, we assert that the agile approach inherently can promote the development of non-technical professional skills that employers desire.

II. Agile methodologies

The word "agile" comes from the Latin word *agilis*, meaning "can be moved easily, light" and the French word *agere* meaning "to drive, to be in motion" [9]. It has gained substantial popularity in the software development domain. At its core, the agile work methodology borrows from the concepts around continuous improvement and system development based on feedback from customers and the environment. Agile teams focus on collaborative planning and goal formulation, leading to more substantial commitment to achieving the goals and delivering the developed system to the customer.

In agile software development, the team agrees on overall goals for delivery of a system (say, an online banking system), and then timebox smaller parts and user stories that they develop towards, then test this with the customer to get feedback on features of the system to enable continuous improvement (such as better user interface) — and enabling adjustment to a changing environment (for example new security challenges).

A well-known agile workflow is Scrum [10], supported by epic stories, user stories, issue poker planning, sprint planning — review — retrospective, daily stand-ups, and other artifacts [11]. Inclusion of the developing team in all aspects of planning and organizing fosters ownership of the system and can improve team cohesiveness.

In [11], the authors created the learning agile methodology, which are guiding principles for when developing courses with the agile methodology:

- ***Adaptability*** *over prescriptive teaching methods.*
- ***Collaboration*** *over individual accomplishment.*
- ***Achievement of learning outcomes*** *over student testing and assessment.*
- ***Student-driven inquiry*** *over classroom lecturing.*
- ***Demonstration and application*** *over accumulation of information.*
- ***Continuous improvement*** *over maintenance of current practices.*

While it is natural to consider agile as a topic and a teaching method in software development courses [12], it can also be applied in other fields such as management or capstone courses [13, 14, 15]. In [16], a list of research opportunities for applying the agile framework in technology education was given, especially addressing effectiveness and other non-technical professional skills. There have also been cases using the Scrum methodology for developing university CubeSats [17], where the students applied Scrum in both the hardware and software development in the project. As the world is becoming more complex and students and educators have to deal with a changing environment and technology daily, having an agile approach to learning and teaching would allow both to change the syllabus easily to include new topics, and to change teaching methods to adapt to changing situations. However, it is important to ensure that courses' adaptions are within the approved learning objectives of a given course, which may limit the adaptability.

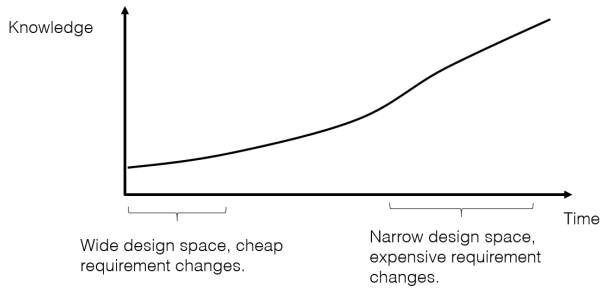

Fig. 1. Illustration of knowledge in product- and software development projects, as described in [18].

III. Knowledge as a product

In new product development projects, a critical catalyst for project success is enabling quick learning. As illustrated in Fig. 1, a significant challenge in product- and software development is that product knowledge is low in the early phases of the project, when the design space is broad, and requirement changes cheap. In the later phases, when the design space is narrow and changes in requirements are expensive [18, 19, 20], the knowledge is high. Therefore, it is essential to quickly increase product knowledge within the development team to avoid costly rework.

Agile methods facilitate quick learning by enabling formative assessment throughout the project [21]. Quick learning facilitation is done externally through continuous feedback from customers and the environment. The formative assessment is also provided internally by having daily stand-up meetings with the project team, monitoring progress, and adapting the project path to externally acquired input [22].

Learning is a process, and by regarding knowledge as a product, agile techniques for formative assessment can be integrated into teaching. The analogy is as follows: the students represent the development team, the learning outcomes represent the requirements, and knowledge is the product. By working in sprints, the students can learn new knowledge and receive frequent feedback throughout the course, ensuring they are on the right track.

In [23], particular challenges with teaching agile in higher education are mentioned. For example, that lecturing during the agile work can be distracting, managing difference in proficiency or skills level, establishing good teams, avoiding group bias and group issues, lack of self-directed learning manifesting in the students, lack of knowledge of agile methods, lack of ownership, and ensuring communication. While these were listed for an industry-supported project course, it can be expected that some degree of these challenges can manifest in any agile-run course.

Based on the authors' experiences with agile methodologies in development and educational settings, such as co-teaching Fuzzy Front End Engineering [24], we suggest a framework for implementing agile formative assessment techniques in teaching and learning. Although using agile in an educational setting is nothing new, our approach focuses on organizing teaching and learning around formative assessment. Here, teaching and learning are organized in full-day sprints focusing on one topic, with small group sizes and trustful learning environments, enabling formative assessment.

We can consider Biggs' Structure of the Observed Learning Outcome (SOLO) taxonomy [25] when building the course, going from uni-structural, to multi-structural, to relational, and finally to extended abstract (even though the course description does not require the latter). The process is analogous to improving the fidelity of a product or a system, and this can be used to divide the sprints and releases of the product.

IV. Applying the framework

To use an agile approach for formative assessment, we start with the skills the candidates shall have at the end of the course. These are analog to the product requirements of a system in a software company. If we here assume that the course can loosely be grouped into a number of parts that build on each other in successive order, we get macro-releases of our knowledge product that each should correspond to an individual part. Let us that assume a course has three overarching parts: X_1, X_2, X_3. For example, the first sprint's goal will be to demonstrate uni-structural knowledge of X_1 topics, the second sprint to demonstrate multi-structural knowledge of X_1 topics, and the third to demonstrate relational knowledge of X_1 topics. Similarly for X_2 and X_3, the last two sprints could for example demonstrate relational knowledge and extended abstract knowledge of all topics together.

The framework, here demonstrated by sprint 2, can be divided in two.

Part one: Theory (total 2.5 hours)

- 15min quiz on last week's topic. The quiz will require the candidate to demonstrate to other classmates that they can *identify, name or utilize* the X_1 topics.
- Summary on result of last week's quiz. Checking that all candidates are familiar with the X_1 topics. What was learned, what was missed (based on results from last weeks quiz), what could be looked further into.
- One-hour traditional lecture with an introduction to how to analyze, apply, compare and relate the topics.

Part two: Practice (total 4–5 hours)

- 5 minute student presentation on last week's project work.
- Project work relevant to the covered topics and course
- 5 minute summary of the day.

A. Course example

Consider an introductory course on microcontrollers where the Arduino [26] platform is used to familiarize new engineering students with coding, electronics and design of cyber-physical systems. As the course is introductory and taught in the first semester, one must assume that the students have no prior knowledge of any relevant domains. The following knowledge and skills could be considered reasonable outcomes from an introductory course on microcontrollers:

- Know the use of microcontrollers as a central component in Internet of Things (IoT), and can describe typical components and architecture, applications, and limitations.
- Design and build simple cyber-physical systems consisting of microcontrollers, sensors, actuators, circuits, and components.

The teaching should be organized in sprints, having a small group size and a trustful learning environment for applying the framework. Moreover, to view knowledge as a product, the students must be familiar with the learning outcomes (product requirements). A useful introduction could be to demonstrate a cyber-physical system during the first session, for example by showing a self-built low-fidelity robot, which is an objectified representation of the learning outcomes. As demonstrated in [8], low-fidelity robots, including software, hardware, and electronics, can be built by students with little or no prior experience in a few weeks. The framework is applied as previously described.

Part one: Theory

- 15 min quiz on last week's topic. E.g. introduction to cyber-physical systems.
- Summary of result of last week's quiz.
- Two-hour lecture on introduction to programming with microcontrollers: basic functions, object-oriented structure, running scripts.

Part two: Practice

- 5 minute group presentation on last week's project work
- Lab work: programming tutorials
- 5 minute summary of the day.

B. Curricular coherence

By implementing the Agile methodology in teaching, the teacher can, in shorter cycles, determine the students' levels of knowledge and what they are struggling with in order to provide feedback. We propose to reiterate old material is to build on it where applicable. For example to emphasize relations between previous and current syllabus and reuse tools and other material or immaterial lab utilities to the extent possible. With a computer or automation engineering programme as an example, this practice would be to stick to one programming language that can be used in all the courses where the students are writing code. The same applies to the choice of scripting languages, such as Matlab and Python, and hardware platforms. The motivation behind this is to develop relations between courses, reduce compartmentalization, and allow students to learn and apply new knowledge on familiar platforms.

This requires that teachers come to terms with a specific set of tools and utilities across the various courses, which might be difficult to organize, for courses and programmes that are well established. Furthermore, many of the programme courses

for automation engineering are practical or can have projects where the theory is applied through practical project work. The course described previously using microcontrollers was intended as an introduction to both electronics and programming where Arduino was utilized. This modular platform can be reused for an electronics course as it is both open hardware and open software, which allows students to either see how it is made or attempt to expand on it. Similarly for object-oriented programming or similar courses, students can be introduced to the Raspberry Pi platform [27], a micro computer that can run an operating system. Raspberry Pi can be reused in many possible elective courses in an automation engineering programme where hardware can be used to demonstrate theory, such as in topics in machine vision, real time computer science or cybernetics. This would give students a minimum number of platforms on which they can build their knowledge and combine the topics in different courses.

More theoretical courses such as mathematics or physics, where the students do not build systems in the traditional sense, could employ computer tools or scripting languages to make something happen on screen, as opposed to manual computations on paper. This could both help the students visualize what is happening and hopefully bring mathematics and physics to a relational level on the SOLO taxonomy when building on a familiar platform.

V. Reflections on Agile in teaching

Agile as a methodology for teaching can improve upon non-technical professional skills by challenging students to work in teams and repeatedly present their work throughout the semester. However, the methodology, as applied to teaching, and consideration of knowledge as a product, has some pitfalls. Knowledge is a decaying product and students may forget parts of the syllabus within the semester or in span of semesters. Furthermore, if the expected student knowledge is behind schedule, the teacher can not reiterate the material with the same ease as normal product development cycles. The teacher is on a one semester-long development pipeline.

Ideally, the class should be divided into groups for the project work, which should be established for the semester. Besides cooperation in project work, the group will serve as the environment for learning outside the classroom. If a student falls behind, it is the student's responsibility to acquire the missing knowledge and group's responsibility to aid in this. This way, a "lagging" student gets help, and the other students in the group get to reiterate the material and attempt to explain it, which can improve their understanding.

The key to mastering the material starts with understanding how to apply the knowledge. As such, the project work should be practical exercises or lab work where possible to facilitate hands-on learning and reiterate previous knowledge. This may increase the student's understanding of the material to the multistructural level of the SOLO taxonomy. Furthermore, during the project work presentations in the following week, the teacher should challenge students to reflect and discuss in plenary. A learning environment where students feel safe is important for these discussions. Additionally, discussions should aim for lifting students to the higher tiers of the SOLO taxonomy and invite the students themselves to, in a constructive manner, analyze and evaluate their work and that of their peers.

Lastly, it might not be easy to adapt the syllabus based on findings later in the semester, this might even be be prohibited by local policies. The syllabus can be considered a contract with learning outcomes that describes what is expected that the students learn. However, *how* the students achieve the learning outcomes (e.g. group sizes, how many lab exercises) does not need to be specified in detail in the syllabus, allowing for adjustments during the semester. There will always be variations between students' ability and the time required to learn and retain new knowledge. While not all students are able to build their knowledge according to the learning outcomes, the teacher should try to the extent possible to guide the student to attain the knowledge required for both future courses and career.

VI. Concluding remarks

Formative assessment is intended for guiding the students' learning during the knowledge acquisition process. A holistic, agile framework for including formative assessment in higher engineering education has been proposed, using examples from a new bachelor's programme: "Automation and Intelligent Systems".

The proposed framework for formative assessment is inspired from Agile software- and product development and requires organizing teaching in sprints and keeping the class size to a manageable number. We realize that these requirements make the framework inapplicable for large courses, such as introductory courses held across study programmes at major campuses, but suitable for smaller campuses or for specialized courses later in the study programmes at larger campuses.

Formative assessment is given through 15-minute quizzes during each session (teacher feedback), short 5-minute presentations on last weeks topic (teacher and peer feedback), and from performing project work in a dedicated place and time (teacher and peer feedback). In addition to receiving feedback throughout the semester, allowing both the students to adapt their learning techniques, and allowing the teacher to adapt teaching techniques, the students are taught a framework for life-long learning that is useful for their engineering practice.

By regarding knowledge as a product and employing a variety of assessment methods (written, reflective, presentation), students will also learn non-technical professional skills that can easily be applied to an engineering work setting, for example when developing software or hardware in new product development project teams. In addition to learning-how-to-learn in teams, by using sprints or cycles, encouraging formative assessment from peers, the framework allows students to benefit from honing other necessary non-technical professional skills, such as effective communication, team work, and time management.

The notion of how exams and grading can be integrated into the proposed agile framework for teaching have not been discussed in this paper. However, each sprint will demonstrate the students' learning and knowledge, and as such a standard four-hour exam should not be necessary to determine their knowledge. The teacher could note what they have not learned during the sprint and focus on these topics during an exam. Further work on the subject in this paper should investigate how grading can be incorporated into the proposed agile framework for teaching.

REFERENCES

[1] A. Kohn, "The Case Against GRADES," *Educational leadership: journal of the Department of Supervision and Curriculum Development, N.E.A*, vol. 69, pp. 28–33, Nov. 2011.

[2] P. Black and D. Wiliam, "Developing the theory of formative assessment," *Educational Assessment, Evaluation and Accountability*, vol. 21, no. 1, p. 5, Jan. 2009.

[3] I. Clark, "Formative assessment and motivation: Theories and themes," *Prime Research on Education*, vol. 1, p. 10, May 2011.

[4] K. Schildkamp, F. M. van der Kleij, M. C. Heitink, W. B. Kippers, and B. P. Veldkamp, "Formative assessment: A systematic review of critical teacher prerequisites for classroom practice," *International Journal of Educational Research*, vol. 103, p. 101602, Jan. 2020.

[5] J. Hattie and H. Timperley, "The Power of Feedback," *Review of Educational Research*, vol. 77, no. 1, pp. 81–112, Mar. 2007, publisher: American Educational Research Association.

[6] A. Gagatsis, "Compartmentalization in Learning," in *Encyclopedia of the Sciences of Learning*, N. M. Seel, Ed. Boston, MA: Springer US, 2012, pp. 665–668.

[7] M. Shah, L. Grebennikov, and C. S. Nair, "A decade of study on employer feedback on the quality of university graduates," *Quality Assurance in Education*, vol. 23, no. 3, pp. 262–278, Jan. 2015, publisher: Emerald Group Publishing Limited.

[8] K. B. Slåttsveen, M. Steinert, and K. E. Aasland, *Increasing Student Confidence and Motivation in a Project-based Machine Construction and Mechatronics Course*. The Design Society, 2016, ISBN: 978-1-904670-80-3.

[9] Merriam-Webster Dict. (2020) agile. [Online]. Available: https://www.merriam-webster.com/dictionary/agile

[10] L. Rising and N. S. Janoff, "The scrum software development process for small teams," *IEEE Software*, vol. 17, no. 4, 2000.

[11] T. C. Krehbiel, P. A. Salzarulo, M. L. Cosmah, J. Forren, G. Gannod, D. Havelka, A. R. Hulshult, and J. Merhout, "Agile Manifesto for Teaching and Learning," *Journal of Effective Teaching*, vol. 17, p. 22, 2017.

[12] G. V. Madhuri and L. N. S. Prakash Goteti, "Adopting agile values in engineering education," in *2018 IEEE 6th International Conference on MOOCs, Innovation and Technology in Education (MITE)*, 2018, pp. 103–106.

[13] J. H. Sharp and G. Lang, "Agile in Teaching and Learning: Conceptual Framework and Research Agenda," *Journal of Information Systems Education*, vol. 29, no. 2, p. 45, May 2018.

[14] Z. Masood, R. Hoda, and K. Blincoe, "Adapting agile practices in university contexts," *Journal of Systems and Software*, vol. 144, no. Special Issue on Software Engineering Education and Training, pp. 501–510, 2018.

[15] A. Ciupe, S. Meza, R. Ionescu, and B. Orza, "Practical agile in higher education: A systematic mapping study," in *2017 XXVI International Conference on Information, Communication and Automation Technologies (ICAT)*, 2017, pp. 1–6.

[16] M. C. Benton and N. M. Radziwill, "A path for exploring the agile organizing framework in technology education," in *2011 Agile Conference*, 2011, pp. 131–134.

[17] N. Garzaniti, S. Briatore, C. Fortin, and A. Golkar, "Effectiveness of the scrum methodology for agile development of space hardware," in *2019 IEEE Aerospace Conference*. IEEE, 2019.

[18] S. Thomke and T. Fujimoto, "The Effect of "Front-Loading" Problem-Solving on Product Development Performance," *Journal of Product Innovation Management*, vol. 17, no. 2, pp. 128–142, 2000.

[19] J. K. Liker and J. M. Morgan, "The Toyota Way in Services: The Case of Lean Product Development," *Academy of Management Perspectives*, vol. 20, no. 2, pp. 5–20, May 2006, publisher: Academy of Management.

[20] T. Welo, "On the application of lean principles in Product Development: a commentary on models and practices," *International Journal of Product Development*, vol. 13, no. 4, pp. 316–343, 2011.

[21] X. Bai, M. Li, D. Pei, S. Li, and D. Ye, "Continuous delivery of personalized assessment and feedback in agile software engineering projects," in *Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training*. Association for Computing Machinery, May 2018, pp. 58–67.

[22] J. Babb, R. Hoda, and J. Nørbjerg, "Embedding Reflection and Learning into Agile Software Development," *IEEE Software*, vol. 31, no. 4, pp. 51–57, Jul. 2014, conference Name: IEEE Software.

[23] K. Lundqvist, A. Ahmed, D. Fridman, and J. Bernard, "Interdisciplinary agile teaching," in *2019 IEEE Frontiers in Education Conference (FIE)*, 2019, pp. 1–8.

[24] K. Slåttsveen, C. Kriesi, M. Steinert, and K. E. Aasland, "Experienced from a positivistic way of teaching in the fuzzy front end," pp. 694–699, 2018, 20th International Conference on Engineering & Product Design Education.

[25] J. B. Biggs and K. F. Collis, *Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome)*. Academic Press, 2014.

[26] (2020) Arduino. [Online]. Available: https://www.arduino.cc/

[27] (2020) Raspberry pi. [Online]. Available: https://www.raspberrypi.org/