Agile and Adaptive Learning via the ECK-model in the Software Development Academy

Richard Glassey, Philipp Haller, and Mattias Wiggberg

KTH Royal Institute of Technology
Stockholm, Sweden
{glassey,phaller,wiggberg}@kth.se

Abstract. This paper reports the learning management experiences within an intensive three-month education that helps newly arrived in Sweden find work as IT professionals. The creation of the Software Development Academy was motivated by the migration crisis and the wider need to help integrate newcomers into the social and professional landscape. Despite having relevant skills and training, many have had their studies and careers disrupted either by conflict, or simply by lacking the profile and networks needed to restart their careers in a new country. With limited resources and time, combined with the intensive pace and diverse student backgrounds, the program faces many challenges that threaten its success. To mitigate these challenges, an agile and adaptive approach was adopted that employs TEL techniques and pedagogical concepts to ensure the program is continuously improving via short iterations and tight feedback loops. The program has just finished its third offering and has continuously improved through weekly collection of knowledge, confidence, and experience data that guide interventions and reactions as they are needed. The experience, process, and model presented here may inspire and benefit other courses with similar profiles and challenges.

Keywords: Lifelong learning · Vocational training · Technology-enhanced orchestration of learning.

1 Introduction

The shortage of IT developers is a substantial challenge for Swedish companies. According to calculations from the Swedish branch organization, "IT & Telecom companies," there is a shortfall of 70,000 software developers in Sweden in 2017 [4], and this lack of IT resources is threatening business growth. This is also the case for the European Union as a whole, but with more dramatic numbers on the excess demand [6]. While there is a significant shortage of workers in this area, at the same time there is a high unemployment rate among newly arrived in Sweden (through voluntary or involuntary migration) [14]. Some of them have university degrees or similar industry experiences on their CVs. Usually, the time from getting a permanent residence permit to finding a job at the right skill level

is counted in years. This creates a demotivating situation for newly arrived to Sweden and a less effective use of workforce resources for the society.

This is where the Software Development Academy (SDA), a joint initiative between KTH Royal Institute of Technology (a university) and Novare Potential (a recruitment consultancy), contributes to both these challenges. For people with the right prerequisites it is possible to carry out an intensive training with about 500 hours of programming teaching and practice during a relatively short period of time, that is, 3–5 months. This is sufficient to meet the skills requirements for a large part of the services where Swedish companies currently have resource shortages. The overall aim with the project is to prepare newcomers to Sweden to become employable in the rapidly expanding IT sector in Sweden and give them the possibility to be employed in a sector where the demand is high. To make sure that the education is relevant for the needs of the job market, the aim is also to involve a number of employers in the project, making sure the contents of the education fit their needs, and thereby increasing the chances of employment directly after the education. The education part of the project is organized and implemented by teachers from the ordinary undergraduate and Master's programs in computer science. In parallel with the IT training, an extensive job matching is conducted by the recruitment consultancy. The candidates are also provided with training in soft skills and workplace culture.

The remainder of the paper will describe the SDA program in brief, before describing the main contribution of a model that enables the program to be both agile and adaptive, despite the challenges of intense pace of delivery and a diverse cohort of students. The model is both lightweight in technology requirements as well as data gathering instruments, and as such could be readily adopted in a similar course that faces these challenges.

2 Software Development Academy

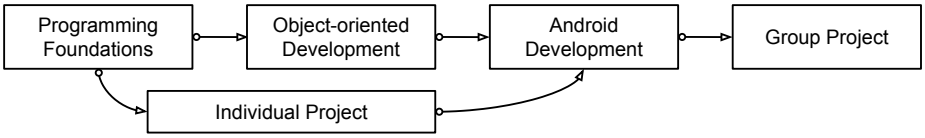


Fig. 1: The structure of the SDA program in terms of the core modules

The SDA program is structured into a sequence of linked modules (e.g., programming foundations, object-oriented design, android development, as well as two project modules), as shown in Figure 1. Every week there are daily lectures and exercises. Each full day typically contains one two-hour lecture given by the module leader, followed by labs that are continuously staffed by teaching assistants. As such, there is little room to develop modules from scratch, and

instead, existing modules from the traditional degree education are repurposed, reduced and reused. While this helps with the efficiency, it does mean that the normal time given for a new topic to be assimilated into a student's knowledge base is limited. This can result in students falling behind with the topics.

Consequently, there is a greater need to continuously monitor the pace, quality and experience of each module as it is delivered, as there are no opportunities for remedial action to occur before the next module begins. Furthermore, the composition of the student cohort is very diverse. Some come with computer science qualifications already, others only come with a passion and interest to learn the subject. In amongst this, the students are typically older and most often having family and housing concerns that typically are associated with newcomers to Sweden. Given the intensity and diversity, there is a pressing need to provide immediate opportunities to adjust the pace of course, provide reviews, whilst letting students express their confidence and test themselves on a regular basis.

The SDA project draws inspiration from the use of agile methods, as a strategy to remain adaptive to the diverse needs of the students as well as a means to continuously improve the quality of the program despite its intensity. Agile methods have had a profound effect upon software engineering [1]. At their core they accept that change is inevitable, and through continuous communication, feedback loops, rapid prototyping and short iterations, the sum of all the incremental developments will have a better chance at delivering a successful solution [3]. The agile philosophy has also spread well beyond the software engineering context, most prominently within project management across many differing domains [13]. Within the domain of education, agile methods have been adopted within software engineering courses, where there is an obvious mapping to the types of methodologies and projects that students develop [11]. However, the agile approach has been less used at the course development level [7].

In many ways, traditional education design (following an ADDIE approach [5]) maps onto the waterfall model: (1) the course is specified according to intended learning outcomes (*requirements*); (2) the syllabus of subjects is planned (*design*); (3) the teaching materials are produced (*development*); (4) the materials are delivered to students (*implementation*); (5) the course is evaluated by students (*testing*); and (6) any changes from the evaluation are incorporated before the next round (*maintenance*), and as such there is potentially some benefits to be gained from a more agile inspired approach. Whilst software projects have a well-defined product, it is harder within an educational context to clearly identify product. However, it is easy to view students as the *clients*, who can play a more active role in the evaluation and development of their education. Traditionally, students perform a course evaluation that feeds into future course improvements [15]. This does not directly benefit these students, but it does provide valuable insights into their learning experience. Clearly, if this evaluation can be integrated into the course at multiple points, students may actually see the benefits more immediately. Another valuable practice is for students to regularly reflect on their own learning [2]. This can act as another proxy for the product of education. The act of determining one's confidence can be a useful

guide to where more focus is required at the individual level. Taken across all students, wider deficiencies can be identified and improvements can be made.

Finally, formative assessment provides more opportunities to evaluate learning, and provides more feedback interventions to help guide student progress, rather than summative assessment [12]. The use of smaller and more regular quizzes on recent topics can help learners discover their own deficiencies and remedy them, whilst signal to teachers the areas that need more attention to correct misconceptions or provide more support. Whilst a product in learning may be difficult to clearly identify, there are meaningful proxies, such as experience of learning, reflections on confidence, and feedback on formative assessment that have been shown to be important in quality improvement of education.

3 ECK Model

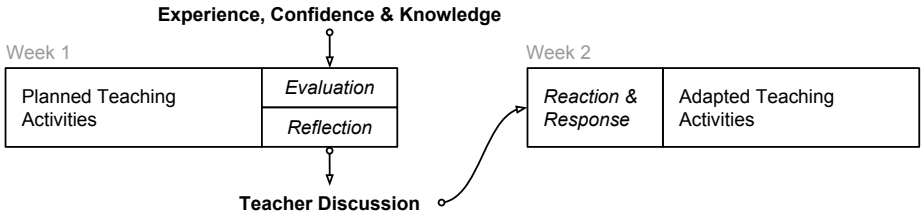


Fig. 2: The ECK-model process across a two-week period

The experience, confidence and knowledge (ECK) model was developed to ensure that the SDA program could be agile and adaptive, despite its intensity and diversity challenges. The first aspect of the model is the order and timing of key events. Figure 2 shows the sequence of events over a typical two-week period. In the first week, the course proceeds with the planned teaching activities. At the end of the week, thirty minutes are devoted to running an evaluation session. It consists of two short surveys that evaluate student *experience* and *confidence*, and a longer quiz to evaluate their *knowledge*. Students receive automatic feedback on the knowledge quiz as soon as they have submitted. A teachers meeting follows the student evaluation to discuss the results. The survey and quiz software (all developed using Google Forms) visualises the results immediately for quick and convenient analysis. This meeting allows teachers from all modules to understand the current situation, and plan changes for the next week based on the data. At the beginning of the next week, the first session presents the results of the surveys and quiz, and also explains the planned response, such as reviewing particularly challenging topics, or adjusting the pace of delivery to either speed up or slow down. Finally, any common trends from the open comments section of the experience survey are addressed during this session.

Table 1: Experience and confidence surveys.

(a) Experience survey question set.

Q1	The pace of the course is manageable at this stage.
Q2	The level of course content is within my ability.
Q3	I feel comfortable in the teaching environment.
Q4	I would like more help with particular topics.
Q5	The course is challenging in a stimulating way.
Q6	I understand what the teachers are talking about.
Q7	I am able to learn from concrete examples that I can relate to.
Q8	The intended learning outcomes help me understand what I am to achieve.
Q9	The course activities help me to achieve the intended learning outcomes.
Q10	My background knowledge is sufficient to follow the course.
Q11	I am able to learn by collaborating and discussing with others.
Q12	I am able to get support if I need it.

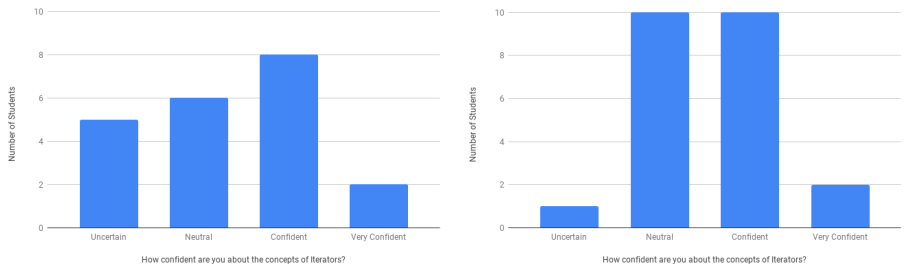
(b) Confidence survey topics for the first four weeks.

Week 1	Week 2	Week 3	Week 4
Class definition	Collections hierarchy	Basic Git	Exceptions
Constructors	ArrayList	Merge conflicts	Try / catch block
Methods	Generic types	**Iterators**	Checked vs unchecked
Types of variables	Array types	Streams	IO Streams
Conditionals	For-loop statements	Coupling	IO Buffers
Boolean expressions	**Iterators**	Cohesion	Inheritance
Method header	Documentation	Code duplication	Extends keyword
Variable data types	Javadoc tool	Purpose of testing	Super keyword
Return values		Testing limitations	Overriding methods

The first component of the ECK model is the *experience survey*. This is a set of twelve questions that remain the same each week, as well as an area for open comments. This is a modified version of the Learning Experience Questionnaire developed for all course evaluations at the host university. It is an evidence-based tool for course evaluation and analysis that helps to examine students' learning experience based on a number of factors that have been found to improve student learning in higher education [8]. Table 1a lists the twelve questions that are used in the experience survey. Question responses are modelled using a five-point Likert scale, from strongly disagree (-2) to strongly agree ($+2$).

The second component is the *confidence survey*, which comprises a dynamic list of topics taught in the current week. Table 1b shows the topics from the first four weeks of a module. Responses are modelled using a five-point Likert scale, from very uncertain (-2) to very confident ($+2$). The highlighted **Iterators** topic appears twice. This is not a mistake; instead, it reflects how this topic had to be revisited in the subsequent week based on the results of the former week's evaluation, and demonstrates the adaptation in action. Figure 3 shows the confidence results on this particular topic, with a clear shift in confidence towards more certainty. Although in effect minor, it nevertheless reflects how an issue in confidence was detected and some immediate actions could be taken to

generate improvements in student confidence. Making this intervention as early as possible avoids the chance that any topic is left as uncertain for any length of time, and learning activities are modified immediately rather than leaving it until later to uncover and address the issue.



(a) Week 2 Results

(b) Week 3 Results

Fig. 3: Shift in the confidence results on Iterators from Week 2 and the results from Week 3 after detecting the issue and planning to revisit the topic at the start of Week 3.

The third component is the 20-minute *knowledge quiz*. It contains 10-20 multiple choice questions. An example quiz can be accessed here: Week 4 Quiz. Preparation of the knowledge quiz takes significantly more effort than the experience and confidence surveys. However, after one iteration of the program, question banks form and speed up subsequent offerings. In line with online assessment, like CMU's Open Learning Initiative [10], special attention is given to feedback. The quiz tool allows feedback to be generated for correct and incorrect answers, as well as a final feedback irrespective of answer. On submission students receive immediate feedback, and later in the program quizzes can be taken again for revision purposes.

4 Admission Data, Diversity and Job Rate

Participant data in Table 2 shows some interesting differences from the normal experience at IT programs in Sweden in general and at KTH in particular.[1] The SDA project has more freedom to innovate in the selection process of candidates, with the result that participant statistics differ from the normal distribution of seats at IT programs within Sweden. In particular, percentages of women normally tend to range between 15-20 percent in IT programs at Swedish universities. Across three iterations of SDA, the share of female participants has been above 40 percent. The share of female participants who manages to finish the

[1] The reader should bear in mind, though, that the data set is limited and hence the statistics are merely descriptive.

course is even higher, which is a result in accordance with more general statistics on university studies in Sweden. The diversity in the classes is obviously high with 9-21 nationalities represented. Syria is the most represented country among the participants, followed by India and Pakistan. The number of participants getting a job after the program was 86 percent after 5 months of the completion of SDA1 (i.e., the first offering of the SDA program). The same figure for SDA2 is slightly worse with 61 percent (this percentage has increased during the last months and more participants are in interview processes). For SDA3, which ended just a month ago at the time of writing, the figure is already 50 percent. Relative to the date of completion it is the best rate of all offerings.

Table 2: Participant data from SDA1-SDA3.

Iteration		Diversity of nationalites	Female participants % of total		In jobs after SDA % in job / finished		
Start	Finish		Started	Finished	Female	Male	Total
Jan/17	Apr/17	9	42.31%	50.00%	81.82%	90.91%	86.36%
Sep/17	Dec/17	21	43.75%	46.43%	69.23%	53.33%	60.71%
Feb/18	May/18	13	45.45%	38.89%	57.14%	45.45%	50.00%

5 Experience and Lessons Learnt

The ECK model attempts to compress good pedagogical practice, student evaluation of learning [15], reflection on learning [2], and assessment with immediate feedback [12,9], whilst not becoming too demanding on teachers or students. In terms of positive experience, the use of Google Forms for capturing and analysing weekly data was a critical success. The unified quiz/survey interface helped reduce production effort. Also, the convenient summarisation and visualisation feature meant that analysis was immediate. This supported the process of directly addressing results as soon as the evaluation ended. Although this simple toolset is not ground-breaking itself, it however mimics the types of low-tech solutions that are often adopted in agile practices. It is not the complexity of the technology that matters; it is the simplicity, speed, usability that encourages teachers to implement the pedagogical interventions described in this work.

For students, regular evaluation was popular to test learning, as well as an outlet to vent frustrations with pace or the environment. For teachers, regular evaluation provided a focus on how to improve the program in real time. On the negative side, the quiz consistency varied across modules, due to the preparation required; modules with multiple iterations could draw previous questions, whilst the development cost for new teachers was a barrier to consistent delivery. Crucially, there was no significant technical burden in using the Google Forms interface to implement ECK components for each module.

With regard to the inspiration drawn from agile methodologies mentioned earlier, the general sense from teachers was that the regular feedback from students in terms of their experience, confidence and knowledge acted like a discussion with clients of a project. It provided a focus on where to make improvements and in a sense, deliver on that requirement. This had the knock-on effect that there never were any surprises at the end of a module or an entire iteration. The commitment to continuous improvement ensured that problems were eliminated quickly, or at the very least they were recognised and discussed immediately with all the participants in the project. Within agile methods, change is welcome and to be expected, and in our experience, taking inspiration within a learning context helped to breakdown the rigidity of the curriculum and schedule and become more flexible to the actual experiences of learning.

Ultimately, the process and model presented in this paper worked as intended – there was continuous communication through feedback loops with the students; each week was a new iteration, and topics were treated as prototypes that could be improved based on the feedback. The agile approach also created closer connections with students, and may inspire other intensive courses to adopt these techniques to remain responsive and adaptive. This will become more important in the future, as life-long learning environments will be populated by diverse cohorts of students, and the experiences within this program can act as an example case, with potential solutions towards adaptive learning.

References

1. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D.: Agile Manifesto (2001)
2. Boud, D., Keogh, R., Walker, D.: Reflection: turing experience into learning (1985)
3. Dybå, T., Dingsøyr, T.: Empirical studies of agile software development: A systematic review (2008)
4. von Essen, F.: IT-kompetensbristen. En rapport om den svenska digitala sektorns behov av spetskompetens. Tech. rep., IT&Telekomföretagen (2017)
5. Gagné, R.M., Briggs, L.J., Wager, W.W.: Principles of instructional design. Performance Improvement **44**, 44–46 (2005)
6. Hüsing, T., Korte, W.B., Eriona, D.: e-Skills in Europe. Trends and Forecasts for the European ICT Professional and Digital Leadership Labour Markets (2015-2020). Tech. rep., empirica (2015)
7. Kamat, V.: Agile manifesto in higher education. In: Proc. 4th Int. Conf. on Technology for Education, T4E 2012. pp. 231–232 (2012)
8. Kember, D., McNaught, C.: Enhancing university teaching: lessons from research into award-winning teachers (2007)
9. Kulkarni, C., Klemmer, S.R.: Learning design wisdom by augmenting physical studio critique with online self-assessment. Tech. rep., Stanford University (2012)
10. Lovett, M., Meyer, O., Thille, C.: JIME-the open learning initiative: Measuring the effectiveness of the oli statistics course in accelerating student learning. Journal of Interactive Media in Education **2008**(1) (2008)
11. Rico, D.F., Sayani, H.H.: Use of Agile Methods in Software Engineering Education. AGILE Conference **0**, 174–179 (2009)

12. Sadler, D.R.: Formative assessment and the design of instructional systems. Instructional Science **18**(2), 119–144 (1989)
13. Serrador, P., Pinto, J.K.: Does Agile work? - A quantitative analysis of agile project success. International Journal of Project Management **33**(5), 1040–1051 (2015)
14. Statistics Sweden (Statistiska centralbyrån): Arbetskraftsundersökningarna (AKU) 2017. Tech. rep. (2018)
15. Wachtel, H.K.: Student Evaluation of College Teaching Effectiveness: a brief review. Assessment & Evaluation in Higher Education **23**(2), 191–212 (1998)