Using Essence in a Software Development Methodologies Course: An Experience Report

Jan-Philipp Steghöfer
Software Engineering Division
Chalmers | University of Gothenburg
jan-philipp.steghofer@gu.se

***Abstract*—We report on our experience of using Essence as an educational tool in a course on Software Development Methodologies. Students used both the kernel and the language of Essence to discuss processes and endeavours, to define and combine practices, and to plan a Scrum-like process that was then applied in a workshop to build a Lego city. We found that using Alpha State Cards and the associated games helped students get a deeper understanding of the coherence of process elements and that the shared terminology was helpful in discussions. However, we observed students struggling with the idea of a language to describe processes and with what constitutes the kernel.**

***Index Terms*—Software Process Education, Essence, Software Engineering**

I. Introduction

An important objective of software engineering education is to let students understand the principles of applying software development processes to structure their work, coordinate with stakeholders and other teams, and to create customer value in a repeatable way [1]. Students must be able to apply different processes in their professional life and they require the ability to understand the ideas and purposes behind them [2]. However, process knowledge and understanding is strongly connected to experience with working as a development team in an organisational context [3]. In an educational setting, students tend to focus their attention on delivering products rather than applying the process correctly [4]. This impedes learning of process aspects and makes it difficult for teachers to provide meaningful education in these matters.

One problem with teaching software processes used to be a lack of common ground for speaking about the elements of processes, how they are combined to form a coherent, applicable process, and how they relate to important aspects of a process such as stakeholders, the produced software system, or the team. *Situational method engineering* [5] was a step towards modularising processes and allowing parts of them to be reused and combined. The *Software & Systems Process Engineering Metamodel* (SPEM) [6] provided a language to describe processes and its constituting parts. Both of these approaches have, however, never reached broad adoption. In particular, high-quality descriptions of software processes in terms of *method content* as prescribed by situational method engineering modelled in SPEM are few and far between. The *Eclipse Process Framework* (EPF) project used to publish descriptions[1] of Scrum, XP and the OpenUP, but the downloadable content is outdated. At the same time, there is little support for combining different practices from these processes into new processes or tailoring the method content to form a coherent whole. Which practices are needed to create a workable process and which impact process design choices have remains a topic for experienced process engineers.

Essence [7] is an attempt to remedy these shortcomings. It provides a *kernel* — a set of process elements that are common to all endeavours — as well as a language —a simple meta-model similar to SPEM to describe practices and patterns that can be combined into a process. It is supported by a growing body of material, in particular a set of games, e.g., to identify the status of a project or to identify next steps. Essence has been designed from the beginning as an educational tool and to allow students and practitioners alike to explore software processes with the help of clearly defined, easy-to-understand concepts and the support of the kernel.

This paper reports on the use of Essence in a course on Software Development Methodologies. We exemplify the use of Essence in the classroom, describe how students perceive and use it, what they understand readily and what they struggle with, and report on observations and recommendations based on our experience. It thus lends support to other teachers who would like to introduce Essence in their courses and provides pointers to how Essence can be used effectively.

II. "Essence" of Software Engineering

Essence is the result of work conducted within the Software Engineering Method and Theory (SEMAT) community[2] to standardise and summarise the essential elements required to describe, compare, tailor and use software processes. It is described in an OMG specification [8] and detailed in an upcoming book [7] that focuses on Essence's practical use and exemplifies how it can be applied in practice. At the time of writing, supporting materials are available, but spread over different websites[3], some of which require registration before download. They include online games (e.g., the Essence Kernel Puzzler[4]), descriptions of serious games to help understand the

[1] https://www.eclipse.org/epf/downloads/configurations/pubconfig_downloads.php

[2] http://www.semat.org

[3] For instance, https://www.ivarjacobson.com/alphastatecards and http://www.software-engineering-essentialized.com/home

[4] https://puzzler.sim4seed.org/

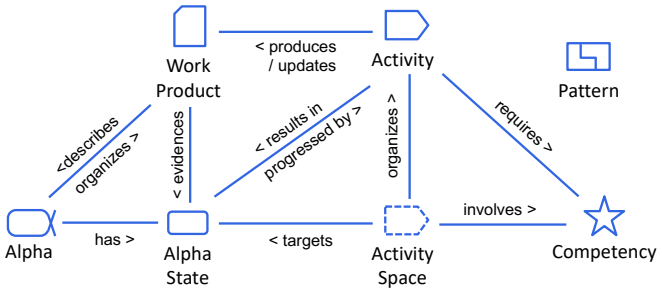

Fig. 1. The Essence language meta-model describes the key elements of essence and how they are connected [7, p. 62]. Patterns and practices can be constructed using these elements.

status of an endeavour, identify goals, or to plan next steps, and templates for cards (see below).

Essence consists of two parts: the kernel and the language. The *kernel* contains recurring, essential parts of software processes. Most notably, it provides *Alphas* (Way of Working, Work, Team, Stakeholders, Requirements, Software System, and Opportunity). These Alphas are the elements of a process that needs to be considered during the endeavour and change state as the endeavour progresses. The states of each Alpha are defined with a checklist. The kernel also contains *Activity Spaces*, i.e., placeholders for the important activities that need to take place in each endeavour. Key *competencies* are defined as well. All of these elements can be categorised using three areas of concern (Requirements, Solution, Endeavour).

The language, in turn, defines a meta-model and a graphical representation for method content and practices. It defines key constructs such as activities, activity spaces, work products, patterns, alphas, and competencies and how these constructs relate to each other (cf. Figure II). The language can thus be used to create complex patterns and practices that can be combined into larger building blocks and full processes.

The *Alphas* play a particular role: using Alpha State Cards, i.e., representations of the different states of the Alphas as playing cards, it is possible to play different games to, e.g., plan an iteration, check the progress of the process, or define milestones and checkpoints. These games are described in an instructional guide [9]. While the cards are available for purchase, a PDF[5] to craft one's own deck is also available.

III. Integrating Essence in the Course

A. Course Design

The course on *Software Development Methodologies* is taught to around 70 second-year students in an international bachelor program on *Software Engineering and Management*. The students have previously been exposed to software processes, in particular to Scrum, in the project courses that run each semester. In these project courses, the focus is on an iterative-incremental way to produce a demoable product at the end of each sprint for review with the teacher based on a high-level introduction to the method. The students have thus not discussed how to construct a process, correctly apply it, and continuously improve it. No learning objectives are associated with processes in these project courses and proficiency in software processes is not examined.

The intended learning outcomes of the *Software Development Methodologies* course cover these aspects. They range from low-level objectives such as "describe the core elements of a software process and the associated method content, including activities, tasks, roles, artefacts, etc." to more advanced topics of software process improvement such as "evaluate a development project, suggest a plan for software process improvement based on the evaluation, and apply the plan." To achieve these intended learning outcomes, the course is based on active learning [10]: students work in groups on different tasks in class after instruction from the teacher and complete assignments that ultimately lead up to an experience report. A typical lecture consists of a quiz at the start to repeat the most important concepts from the previous session, the introduction of new content with a few slides by the teacher, and group work to apply the new concepts. The latter steps are repeated so that two to three new ideas or concepts are covered in each session.

Students also go through two Lego Scrum simulations [11]. In the first of those workshops, students create a shared experience of applying a process and witness issues with the process first hand. In the second workshop, at the end of the course, they apply the software process improvement plans they have constructed during the weeks since the first simulation. The overall structure of the course is as follows:

Part 1: Software Processes

1) Understand software processes, including their basic building blocks and lifecycles.
2) Construct a process to apply for collaboratively building a Lego city.
3) Apply the process for building a Lego city and report on the experiences with this endeavour.

Part 2: Software Process Improvement (SPI)

4) Identify improvement needs from the experience of applying the process.
5) Identify goals, questions, and metrics to formalise improvement needs and make improvements measurable.
6) Construct a process improvement plan using agile practices and method content from prescriptive SPI methods such as CMMI.
7) Apply the improvement plan in a second workshop and report on the experience.

In the final, individual report that constitutes the examination, the students describe and analyse their experience following a similar structure. In addition, they need to relate their own experience to knowledge from guest lectures and the course literature which consists of a collection of papers mainly focused on SPI as well as selected chapters from *Software Engineering Essentialised* [7]. Mandatory assignments cover most of the points in this list. The aim is to guide the students through the steps and provide continuous feedback.

[5]https://www.ivarjacobson.com/publications/cards/alpha-state-cards-pdf-version

B. Use of Essence

Essence is mostly used in the first part of the course when introducing software processes and how to construct them, i.e., during the first two step listed in Section III-A. In addition, students can apply concepts from Essence in step three, e.g., to define milestones for their endeavour or to check their progress, and it is used in step four to check the health of the completed endeavour. Table I lists the different teaching moments in which Essence is used along with the associated intended learning outcomes. At the beginning of the course, each student received a set of Alpha State Cards. For the lecture on Scrum, students also received a set of Essence Scrum cards[6] each.

IV. Observations and Recommendations

Using Essence as an educational tool proved to offer a number of advantages. First of all, Essence defines a clear terminology that is extremely helpful when discussing the concepts of software processes in the classroom. The software processes literature does unfortunately not use a common and clear terminology (witnessed, e.g., in the confusion about the terms "method", "methodology", "approach" and "process" which are sometimes, but not always, used synonymously).

Furthermore, the Alphas have proven useful when discussing the different relevant aspects of a process. The games using the Alpha State Cards are helpful catalysts for discussions. The Alphas make the necessary building blocks explicit and the games cover important parts of engaging with the process. In the classroom, different Alphas and their possible states can be discussed in relation to how certain practices influence them. At the same time, the games give the students tools to plan, monitor, and analyse the endeavour themselves.

Finally, transitioning between Alpha states especially for Team, Stakeholders, and Way of Working can be easily tied to software process improvement. The "Way of Working" Alpha, e.g., describes the process maturity of the development team. In its more advanced states, it contains checklist items such as "Continuously tuned" (state five of six). This corresponds to a continuous approach to SPI in which improvements are discussed and implemented as needed. It is also possible to discuss how the method content students select to improve the process impacts the Alpha states. For instance, if students adopt Kanban boards, this has an impact on the "Work" Alpha and specifically on how tasks are planned and if unplanned work is under control

However, based on feedback from students and the grading of the assignments, there are a number of caveats that a teacher aspiring to use Essence in the classroom can encounter. First of all, the students struggled in using the Essence language and the elements from the kernel as they had insufficient knowledge of modelling and language engineering. The concept of a meta-model and its instantiation was thus not clear to them. They also had issues in understanding how to make the activity spaces more concrete by defining suitable activities within them, partially because they struggled with understanding the purpose and rationale behind the pre-defined activity spaces.

The main issue is the *lack of experience* students have with software processes. That makes it difficult for them to understand the complications that can arise and to make the abstract concepts of software processes concrete for them. Essence helps a little bit in this regard since it names the important aspects in the Alphas and lists the possible issues that can arise in the checklists of the Alpha States. However, many of these issues are difficult to grasp with little experience. In particular for the Alphas "Opportunity" and "Stakeholders", the students are missing a frame of reference. Without experience in requirements engineering, the later stages of the "Requirements" Alpha are difficult for them to grasp. Their poor understanding of teamwork likewise makes it hard to them to understand the meaning of the more advanced states of "Team".

At the same time, Essence can be a supporting tool in alerting students to these facets of software processes that are often overlooked in an educational setting that is rather focused on technical aspects [4]. Since issues are made explicit, an analysis of the Alpha States and their checklists in the classroom can lead to interesting discussions about them. If the course contains practical elements, either in the form of an actual software project or in the form of a simulation, it is also possible to create a shared experience that can be drawn from in the discussions in the classroom.

In summary, a teacher using Essence in the classroom should ensure that the scaffolding provided takes the level of experience of the students into account. Using the language to construct patterns and practices should also be supported by sufficient skills in modelling. Grounding the discussions about software processes in a concrete, practical shared experience and applying Essence in its context has proven to be advantageous.

V. Conclusion

Essence is a helpful pedagogical tool that supports students in understanding the somewhat abstract concepts of software engineering processes. In particular the Alpha State Cards and the associated serious games have proven to be an asset to clarify important concepts of software processes such as structured planning and measuring progress. The Essence language and the kernel are useful to introduce students to process modelling and highlight the most important building blocks of a software process. If the use of Essence is carefully scaffolded and combined with a practical element, it provides many advantages such as a clear terminology, a construction kit for processes, and a set of serious games that show different process aspects and are useful in both education and practice.

An important step to support teachers in the use of Essence in the classroom is to make educational material available in a central location. This effort is currently under way within the SEMAT community and will hopefully provide a repository of teaching concepts, slides, quizzes, assignments, etc. as well as a forum for exchange between teachers.

[6]http://ss.ivarjacobson.com.pages.services/essential-scrum?ts=1528896470457

TABLE I
THE DIFFERENT LECTURES, ACTIVITIES, AND ASSIGNMENTS THAT USE ESSENCE ALONG WITH THEIR INTENDED LEARNING OUTCOMES.

Essence concepts	Activities	Assignment	Intended Learning Outcomes
Lecture 1: Building blocks of a software process (Steps 1 and 2)			
• Essence kernel: areas of concern, alphas and alpha states, competencies, activity spaces • Essence language: language constructs and meta-model	• Select one Activity Space. Define two activities for the selected Activity Space with the the following information: (1) input (in terms of work products); (2) output (in terms of work products); (3) necessary competencies; (4) alpha states changed by the activity • Milestone Mapping Game [9]: Define a number of milestones from inception to delivery. For each Alpha, identify the state that the Alpha should be in when the milestone is achieved.	• Play Progress Poker and Checkpoint Construction [9] for a provided software development scenario and justify the different choices made.	• describe the core elements of a software process and the associated method content, including activities, tasks, roles, artifacts, etc.
Lecture 2: Structured Planning and Progress (Steps 1 and 2)			
• Use Alpha State Cards to play serious games to plan project and measure its progress	• Checkpoint construction • Progress Poker		
Lecture 3: Agile Development Processes (Steps 1 and 2)			
• Model and combine agile practices using the Essence language and elements from the Essence kernel; • introduce iterative-incremental development lifecycle and different approaches to agile software development.	• Build your own practice: Describe one of the practices Test-driven Development, Continuous Integration, or Product and Sprint Backlog using information from the provided sources. Use the Essence language and elements from the kernel. • Kanban: Create a practice "Manage Kanban Workflow" using the Essence language and elements from the kernel. Define the activities that it should include. • XP: Create a model of XP using the Essence language and elements from the kernel.	• Select practices to use in the workshop and provide a rationale. Combine them to a process that addresses all relevant activity spaces. Create one or more diagrams that show the activity spaces, how Alphas and work products are connected, and which patterns are used. • Describe how you are going to scale the process to include other teams and stakeholders. • Describe how you plan to instantiate the process by progressing the "Way of Working" Alpha from "Foundations Established" to the "In Use" state. Remember that you should discuss roles, rules, tools, and schedule.	• describe the core elements of a software process and the associated method content, including activities, tasks, roles, artifacts, etc. • describe and discuss the advantages and disadvantages of different lifecycles, including Waterfall, V-Model, Iterative, Incremental, and their respective combinations
Lecture 4: Scrum (Steps 1 and 2)			
• Important concepts in Scrum, including Sprints, Sprint Reviews and Sprint Retrospectives as well as customer involvement; • scaling Scrum to several development teams using a program and a team level [7, p. 249ff.]	• Use the Essence Scrum cards to answer the following questions: (1) What's in a sprint? (2) How are requirements handled? (3) How do the team patterns impact the activities? • Process instantiation: Discuss what needs to be done to progress to the "In Use" state of the "Way of Working" Alpha!		
Workshop: Use constructed process in a Lego Scrum simulation (Step 3)			
• Process is defined using Essence language and kernel • Alpha State Cards available to students, games known	• Students define activities on their own as part of their process; • in practice Essence was not used in the simulation.		• evaluate a development project, suggest a plan for SPI based on the evaluation, and apply it
Lecture 5: Process Quality (Step 4)			
• Use Alpha State Cards to identify issues in the application of a process and find areas of improvement	• Health Monitoring: track the health of your endeavour and identify what the next steps are using the Alpha State Cards.		• evaluate a development project, suggest a plan for SPI based on the evaluation, and apply it

REFERENCES

[1] A. Heredia, R. C. Palacios, and A. de Amescua Seco, "A systematic mapping study on software process education," in *SPETPSPICE*, 2015, pp. 7–17.

[2] M. Kuhrmann, D. M. Fernandez, and J. Münch, "Teaching software process modeling," in *ICSE-SEET*, May 2013, pp. 1138–1147.

[3] L. McLeod and S. G. MacDonell, "Factors that affect software systems development project outcomes: A survey of research," *ACM Computing Surveys (CSUR)*, vol. 43, no. 4, p. 24, 2011.

[4] J.-P. Steghöfer, E. Knauss, E. Alégroth, I. Hammouda, H. Burden, and M. Ericsson, "Teaching agile: Addressing the conflict between project delivery and application of agile methods," in *ICSE-SEET*. ACM, 2016, pp. 303–312.

[5] B. Henderson-Sellers and J. Ralyté, "Situational method engineering: state-of-the-art review," *Journal of Universal Computer Science*, vol. 16, no. 3, 2010.

[6] Object Management Group (OMG). (2008) Software & Systems Process Engineering Metamodel Specification Version 2.0. [Online]. Available: https://www.omg.org/spec/SPEM/2.0/

[7] I. Jacobson, H. B. Lawson, P.-W. Ng, P. E. McMahon, and M. Goedicke, *Software Engineering Essentialized*. SEMAT, 2018. [Online]. Available: http://www.software-engineering-essentialized.com/

[8] Object Management Group (OMG). (2018) Essence Specification Version 1.2. [Online]. Available: https://www.omg.org/spec/Essence/1.2/

[9] Ivar Jacobsson International, "Alpha state card games – instructional guide," 2015. [Online]. Available: https://www.ivarjacobson.com/publications/brochure/alpha-state-card-games

[10] M. Prince, "Does active learning work? a review of the research," *Journal of Engineering Education*, vol. 93, no. 3, pp. 223–231, 2004.

[11] J.-P. Steghöfer, "Providing a baseline in software process improvement education with lego scrum simulations," in *ICSE-SEET '18*. ACM, 2018, pp. 126–135.