Essence: A Framework to Help Bridge the Gap between Software Engineering Education and Industry Needs

Pan-Wei Ng
Ivar Jacobson International
Singapore
panwei@ivarjacobson.com

Shihong Huang
Dept. of Computer & Electrical Engineering and Computer Science
Florida Atlantic University
USA
shihong@fau.edu

Abstract

Given the time limit, software engineering courses in universities can only emphasize a particular development approach or method; therefore, it is challenging to prepare graduates to face the diverse range of approaches and methods used by industry. One of the issues software engineering education faces is the lack of a framework to understand and compare the similarities and differences among diverse practices used by different companies versus what students learn at school. Software Engineering Methods and Theory (SEMAT) is a new international initiative that bridges the gap among industry, education and research. SEMAT latest result is the submission to OMG standard, namely the Essence – Kernel and Language for Software Engineering. Instead of teaching diverse range of software engineering development methods, Essence provides a novel way of thinking of the different software development methods and approaches. This paper discusses the value of Essence to software engineering education and the preliminary feedback from university professors and lecturers.

1. Introduction

Both universities and industry have invested significant efforts to adequately prepare students. These efforts include designing relevant and balanced software engineering curriculum for students, such as SE 2004 [1] GSWE2009 [2]. Additional efforts also include incorporating software quality assurance (MSwA) [3] and measurement [4]. However, as Boehm [5] noted, software engineering is a rapidly evolving field with new opportunities and challenges. Industry practices must constantly evolve in order to keep up with such demands. Consequently, universities try to follow suit and have revised their courses by shifting their emphases for example from traditional software development approaches to Unified Process [6], recently to agile development approaches [7]. Universities also expand their curricula from a purely technical perspective to include soft skills training such as teamwork [8] [9] [10].

However, given the limited class time and relatively young age of students, there is a limit as to what students can learn from a single software engineering course. We observe that university education usually emphasizes one particular software development approach or method. Explicit attempts to address the differences between what students learn and what industry does is extremely limited, and when they do exist, they are usually rhetoric.

We believe that a fundamental problem is the lack of a framework to compare the similarities and differences among diverse practices used by different companies, what students learn and what their (future) employers require. Consequently, graduates

978-1-4673-5140-9/13/$31.00 © 2013 IEEE CSEE&T 2013, San Francisco, CA, USA

cannot be equipped to systematically understand and express the differences between what their employers' practices versus what they are taught. Without this ability, they have to re-learn from the beginning, albeit using yet another approach and terminology of their employer's.

Software Engineering Methods and Theory (SEMAT) [11], is a new international initiative that bridges the gap among industry, education, and research. One of the results of SEMAT work is known as *Essence– software engineering kernel and language* [12] [13], which will soon become an OMG standard [14]. We believe that Essence provides an attractive framework to deal with the diversity of the many industry practices and to better bridge the gap of software engineering education.

The objectives of this paper are the following:

1. To provide an overview of Essence kernel and language
2. To show how Essence can be used to describe the commonality and diversity of software development methods
3. To identify the value of Essence to software engineering education
4. To report universities' initial feedback on using Essence as a framework for software engineering training.

2. Essence – Software Engineering Kernel and Language

The Essence kernel and language [11] [12] [13] provide a common ground for understanding and describing commonality and diversity of software engineering practices. In particular, the Essence Language models things which software developers work with as *alphas*, which represents different dimensions of software engineering challenges. The Essence kernel identifies a core set of alphas separated into different areas of concerns. These alphas are namely, *Opportunity*, *Stakeholders*, *Requirements*, *Software System*, *Team*, and *Work and Way of Working*.

Each alpha has a series of state progressions that help development teams understand and deal with the risks and challenges for that alpha. For example, the progress of a set of requirements goes through the following states: *Conceived*, *Bounded*, *Coherent, Acceptable, Addressed, Fulfilled* (see Figure 1). The idea is to help teams detect risks in development and to deal with them. For example, if a team attempts to make requirements *Coherent* before getting them *Bounded* (i.e. understanding the boundaries), they may risk wasting their effort if the requirements boundary changes. The Essence specification provides detailed checklist of what each alpha and what each state mean.

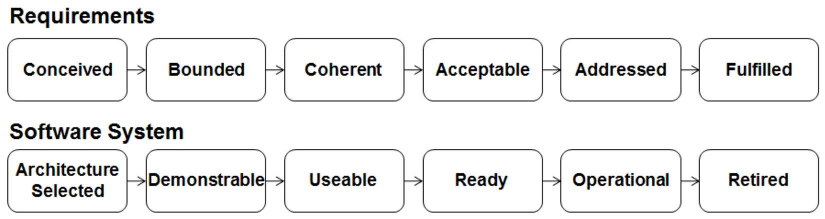

Figure 1. Alphas have progression states

It is important to note that this is not waterfall thinking because a team can have different sets of requirements (instances), each residing within different states. The kernel can deal with large-scale software development, which has multiple sets of requirements (instances), and multiple software systems (instances). Software engineering can then be described as bringing the instances of the alphas from one state to another until the desired software engineering goal is achieved, such as the delivery of a software system. Other than alphas, the Essence also defines other method elements, which can be used to describe practices on top of the kernel.

3. Characteristics of Essence that are of Value to Software Engineering Education

The kernel has several important characteristics that are very useful for conducting real world software engineering endeavors, which are also beneficial to software engineering education. The following sections describe briefly these characteristics and the value they bring to software engineering education – from point of views of both instructors and students. Jacobson et al. [12] [13] describes these in further details with examples.

Actionable – Essence emphasizes what a team needs to do to achieve progress. The alpha states play a central role in what a team does: to help determine the current state of development is and to decide how to move to the next state. This allows instructors to assign work using alpha states and review students' work accordingly. From students' perspective, Essence gives students a way to think about how to achieve progress in their project work, and to report progress to their instructors.

Tangible – Each alpha states and their checklist are presented as a deck of cards (business card size). We usually get project teams to discuss what they need to do by laying out the cards on the table and moving them around [13]. The state definitions can be used to design task boards and Kanbans [15] that are common in agile methods and tooling today.

Configurable and Extensible – Essence provides several mechanisms to describe the diverse range of software engineering approaches. As an example, Figure 2 describes the differences between modern software development and traditional and more conservative development, which has different risk emphasis.

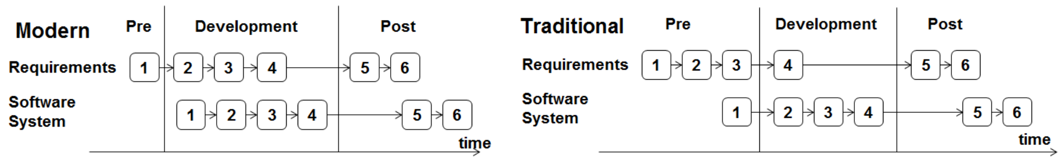

Figure 2. Difference between Traditional and Modern Development Approaches

For simplicity, Figure 2 divides the lifecycles into three phases, namely: pre-development, development and post-development and use the Requirement alpha states and Software System alpha states (numbered in Figure 2) to describe the differences between the two lifecycles. As shown in Figure 2, a modern lifecycle is more responsive and is able to proceed with development at earlier alpha states, whereas a traditional lifecycle tends to be more conservative and waits for more states to be achieved before starting development. Note that the state numbers in Figure 2 correspond to the states in Figure 1.

The Essence also provides a mechanism to extend the kernel by composing practices (e.g. SCRUM, Test Driven Development, etc.) to form a team's own method in a precise manner. This helps students understand the relationships between different practices and how each practice can contribute to project success.

4. Preliminary Feedback from Universities

The first author has conducted both training and coaching of Essence within an industry context, but that is not the focus of this paper. The emphasis of this paper is instead on universities' perceptions of how Essence can help them improve their courseware and help bridge the gap with industry.

We had the opportunity to introduce and gather feedback from 7 of the top 10 universities in China through a workshop conducted in December 2012. The rapid growth of the Chinese software development population represents a major challenge for Chinese universities, which explains their interests in the SEMAT initiative. We divided the participants (professors and PhD. students) into small groups and conducted hands on exercises:

Exercise 1 – We asked participants to identify the challenges software professionals face. Each team gave rather different answers and with different terminology.

Conclusion: Participants immediately saw the importance of having a consensus on terminology, and work such as Essence would be helpful.

Exercise 2 – We asked the participants to map the challenges to the Essence kernel. The participants were able to map the challenges from Exercises 1 to Essence language elements including the alphas, albeit with some help since they were not familiar with Essence yet.

Conclusion: The Essence is able to encompass a wide range of challenges, particularly the typical ones in software engineering. This implies that Essence provides a sufficient breadth of coverage of software engineering disciplines.

Exercise 3 – We asked the participants to describe and compare software development lifecycles of a mission critical project versus that of a mobile application using the alpha states. The participants were able to describe the development lifecycle that is similar to Figure 2 above, although there were some differences in their answers. There was a Eureka moment, when participants saw that they could represent different lifecycles by shifting the alpha states around.

Conclusion: This again demonstrates that Essence provides a tool – a common set of alphas to represent diversity in lifecycles.

Exercise 4 – We asked the participants to describe the scope of the undergraduate curricula using the alpha states. All of the teams indicated that their curricula do not cover all the states, and in particular, alphas such as Opportunity, Stakeholders, Team, and Way of Working were mostly left out by the curricula.

Conclusion: The participants recognized immediately that the business-social aspects of software engineering represent a huge void in their curricula. More importantly, they saw Essence as a way to describe the scope of a curriculum, and the scope of each course.

Participants left the workshop with a great satisfaction and good understanding of Essence and how it can benefit their curriculum. They saw the value of something like Essence as a fundamentally different way to address the gap between education and industry.

5. Conclusions, Limitations and Further Work

Universities cannot teach everything that industry requires. But university can provide students with a firm grasp of the fundamentals and give them the tools to learn and understand the diversity of software engineering later in their career. Essence is such a framework that provides both the fundamentals and way of thinking of dealing with the ever-changing industry needs. Essence is complementary to SE 2004 [1] GSWE2009 [2]. Essence provides a way to glue different practices together.

Certainly, additional work is still needed to introduce a course based on Essence into curriculum within a university's setting. We will be conducting a tutorial, entitled "Essence - Kernel and Language for Software Engineering Practices" in ICSE 2013 [16]. This tutorial will be based on the workshop in China mentioned above. Essence is not just a framework for education; it is also useful for industries and research. There is ongoing work in SEMAT to apply Essence in both areas.

References

[1] Timothy C. Lethbridge, Richard J. LeBlanc Jr., Ann E. Kelley Sobel, Thomas B. Hilburn, Jorge L. Díaz-Herrera. "SE2004: Recommendations for Undergraduate Software Engineering Curricula" in *IEEE Software*. Nov/Dec 2006.

[2] Mark Ardis, Pierre Bourque, Thomas Hilburn, Kahina Lasfer, Scott Lucero, James McDonald, Art Pyster, and Mary Shaw. "Advancing software engineering professional education." In *IEEE Software*. Vol 28, no. 4, 2011, pp 58-63.

[3] Nancy R. Mead and Dan Shoemaker. "Two Initiatives for Disseminating Software Assurance Knowledge" in *Crosstalk The Journal of Defense Software Engineering*. Sep/Oct 2010, pp25-29, 2010.

[4] Mónica Villavicencio and Alain Abran. "Software Measurement in Software Engineering Education: A Comparative Analysis." in *Proceedings of International Conferences on Software Measurement IWSM/MetriKon/Mensura*. 2010, pp. 633-644.

[5] Barry Boehm. "Some Future Software Engineering Opportunities and Challenges." in *The Future of Software Engineering*. S. Nanz, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1–32.

[6] Pierre N. Robillard, Philippe Kruchten, and Patrick d'Astous. *Software Engineering Using the Upedu*. Addison-Wesley Longman Publishing Co., Inc., 2002.

[7] Orit Hazzan and Yael Dubinsky. "Why software engineering programs should teach agile software development." in *ACM SIGSOFT Software Engineering Notes* 32, no. 2 (2007): pp 1-3.

[8] Shihong Huang, Dragutin Petkovic, Kazunori Okada, Marc Sosnick, Shenhaochen Zhu and Rainer Todtenhoefer. "Toward Objective and Quantitative Assessment and Prediction of Teamwork Effectiveness in Software Engineering Courses." in *ACM SIGSOFT Software Engineering Notes* (38:2), January 2013

[9] Dragutin Petkovic, Gary Thompson, Rainer Todtenhoefer, Shihong Huang, Barry Levine, Parab, S., Singh, G., Soni, R., Shrestha, S. "e-TAT: Online Tool for Teamwork and 'Soft Skills' Assessment in Software Engineering Education" in *Proceedings of IEEE 40th Annual Frontiers in Education*, 2010.

[10] Dragutin Petkovic, Kazunori Okada, Marc Sosnick, Aishwarya Iyer, Shenhaochen Zhu, Rainer Todtenhöfer, Shihong Huang. "Work in Progress: A Machine Leaning Approach for Assessment and Prediction of Teamwork Effectiveness in Software Engineering Education." in *Proceedings of IEEE 2012 42nd Frontiers In Education* (FIE 2012 October 3 – 6, 2012, Seattle WA, USA).

[11] Ivar Jacobson, Shihong Huang, Mira Kajko-Mattsson, Paul McMahon, and Ed Seymour. "Semat—Three Year Vision." *Programming and Computer Software*, Volume 38, Issue 1, pp 1-12, Springer January 2012.

[12] Ivar Jacobson, Pan-Wei Ng, Paul McMahon, Ian Spence, and Svante Lidman, "The essence of software engineering: the SEMAT kernel." *Queue*, no. 10 (2012): 40.

[13] Ivar Jacobson, Pan-Wei Ng, Paul McMahon, Ian Spence, and Svante Lidman. *The Essence of Software Engineering: Applying the SEMAT Kernel*. Addison Wesley, 2013.

[14] Essence - OMG Submission online at http://www.omg.org/cgi-bin/doc?ad/2012-11-01

[15] Henrik Kniberg, *Kanban and Scrum-making the most of both*. Lulu.com, 2010

[16] Shihong Huang, Ivar Jacobson, Pan-Wei Ng, Arne J. Berre, and Mira Kajko-Mattsson. "Essence: Kernel and Language for Software Engineering Practices." In *Proceedings of the 35th International Conference on Software Engineering* (ICSE 2013), Tutorial, San Francisco, CA, May 2013.