

Scrum Powered by Essence

June Sung Park

Industrial and Systems Engineering
Department
Korea Advanced Institute of Science and
Technology
Daejeon, Korea
82-10-9107-5661
june.park@kaist.ac.kr

Paul E. McMahon

PEM Systems
Binghamton, NY
1- 607-798-7740
pemcmahon1@gmail.com

Barry Myburgh

Jo'burg Centre for Software Engineering
(JCSE), School of Electrical and
Information Engineering
University of the Witwatersrand
Johannesburg, South Africa
27-11-717-6308
barry.myburgh@gmail.com

DOI: 10.1145/2853073.2853088

<http://doi.acm.org/10.1145/2853073.2853088>

ABSTRACT

This paper shows how Scrum project management practice can be described using Essence kernel and language which has recently been adopted as an official Object Management Group standard for creating and enacting software engineering methods. Practical benefits of using Essence as a common foundation for defining software engineering practices are demonstrated. These practical benefits include the ability to compare practices, assess potential gaps, make needed practice improvements, and assemble select practices into a coherent method to benefit the project team. In addition, by providing practical checklists, as opposed to conceptual discussions, the Essence-powered practice becomes something the team uses on a daily basis. This is a fundamental difference from traditional approaches, which tend to overemphasize method description as opposed to method use.

Categories and Subject Descriptors

K.6.3 [Computing Milieux]: Management of Computing and Information Systems – *software process*

General Terms

Management, Standardization, Languages

Keywords

Essence kernel, Essence language, software engineering method, software engineering practice, Scrum project management

1. INTRODUCTION

Software engineering has evolved since the 1960s from structured methods, to information engineering methods, object-oriented methods, unified methods, then to agile methods, to name just the most well-known ones. According to [1], 32 software engineering methods emerged during the last 20 years. However, there has been no globally agreed, common foundation that is shared across all the software engineering methods. As a result, it is difficult to compare those methods, to learn new methods, and combine them to address the specific need of a particular project.

Essence kernel has been developed by a community of software engineering experts called SEMAT [2], and adopted by the Object Management Group as a standard for providing common elements, language and framework for describing software engineering methods [3]. Essence provides a simple way to compose a method for a specific project from practices written in common notation (i.e., Essence kernel and language) as shown in Figure 1 [3]. A practice here means a systematic and repeatable way of working which addresses a specific aspect of software development effort. Essence can be used to define all types of lifecycle model—from the most lightweight agile lifecycle through more formal iterative lifecycles to the most formal and traditional waterfall lifecycles.

It is important to understand that Essence is not an alternative to any method or set of practices, such as Scrum. Essence contains no practices, but rather it is a framework on which existing or new practices can be described, compared and improved.

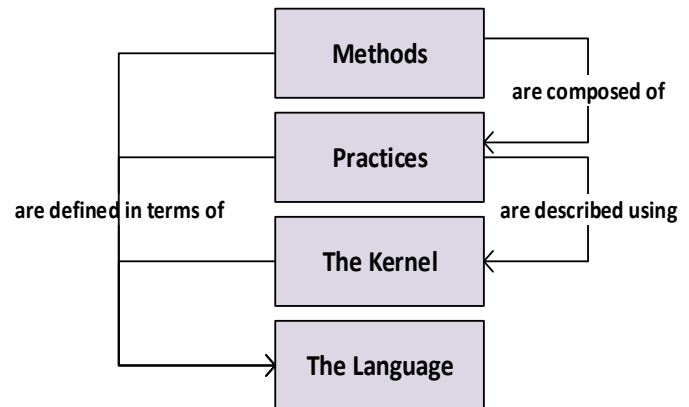


Figure 1. Method Architecture

2. LITERATURE AND CASE STUDIES

The features, applications and values of Essence are discussed extensively in [4, 5]. They showed how Essence kernel can be used to determine where the project state is and where to go next regardless of specific practices employed in the project. [6] discusses how Essence can improve competitive advantages of software businesses—both the software product business and the professional IT service business. It shows how Essence can enable adaptive and agile composition and enactment of software engineering methods in the face of ever increasing volatility and uncertainty of software business environments. [7, 8, 9] discuss how Essence-based project planning and control can add value to agile practices. The added value includes: to allow a project team to compare and select the best practices for the project; to evaluate the completeness of a set of practices selected for the project; to measure the project progress and health; and to easily and safely change the set of practices as the need arises.

A number of case studies have been reported where Essence-based software engineering and project management were applied with success. Red Hat Architect Team have been using Essence to help standardize its approach to engagements [10]. They are using *alpha state cards* in project planning, which provide consistent language and measurable objectives with which to access the current state, or articulate next steps and goals. They created a dashboard website to record and report the current *alpha states* of engagement projects, which facilitated high-level project governance. Essence was also used to build the Red Hat Architectural Framework which provides a framework to collate and index established approaches: techniques, activities, practices and methods. This enables consultants to find suitable approaches when seeking to achieve goals expressed in terms of Essence. Red Hat also applies Essence to analyzing existing methods and developing more rounded practices. This type of analysis has proven useful in finding the

holes in existing processes and reworking a process to support a progressive state-based model that is easier to track and apply.

Fujitsu UK has been using Essence, in particular *alpha state checklists*, in iteration planning with customers [11, 12]. They also used Essence to analyze, compare, standardize and integrate various Design and Build Methodologies (such as Architecture DBM, Applications DBM, Infrastructure DBM, Service DBM, etc.). With all the disruptive changes occurring in IT today including cloud, mobile, big data and IoT, Fujitsu felt they needed a common framework to standardize their methodologies and applied the Essence kernel successfully to that end. The methodologies were further improved so that quality assurance involved checking *alpha states* rather than activity completions. Fujitsu has 165 thousand employees across most geographies in the world. Fujitsu is now working on developing a common set of global standard methodologies and processes using the Essence as the common framework.

The reader is referred to [13, 14, 15, 16] to find how a large insurance company (Munich Re) and a Chinese global telecommunications equipment vendor are applying Essence to their software engineering endeavors. [17, 18, 19] show how a research institution (SINTEF) and universities (Carnegie Mellon University and National University of Columbia) used Essence to improve research and education of software engineering.

3. ESSENCE KERNEL

Essence kernel contains essential elements that should be addressed in every software engineering endeavor. The kernel elements have 3 categories: *alphas*, *activity spaces* and *competencies* [3]. As shown in Figure 2, there are 7 *alphas* (Figure 2A)—*stakeholder*, *opportunity*, *requirements*, *software system*, *work*, *way of working* and *team*—and 15 *activity spaces* (2B) and 6 *competencies* (2C). *Alphas*, *activity spaces* and *competencies* are grouped in 3 areas of concern: *customer*, *solution* and *endeavour*.

3.1 Essence Alpha and State Checklist

Alphas are manifested by *work products*. Each alpha has a set of *states*. As shown in Figure 3, the *opportunity* alpha may be in one of six states over the lifecycle of a software engineering endeavor. If, for example, Product Vision—a work product in Scrum practice—is mapped to the *opportunity* alpha, Product Vision is deemed to go through the states listed in Figure 3. *Essence* also provides a *checklist* for each *alpha state* as illustrated in Figure 3. Cards can be used to recall the kernel elements. Figure 3 shows two types of cards—*alpha definition card* and *alpha state detail card*. Note that the *checklist* in an *alpha state detail card* shows an abbreviated version of the full *checklist* specified in [3].

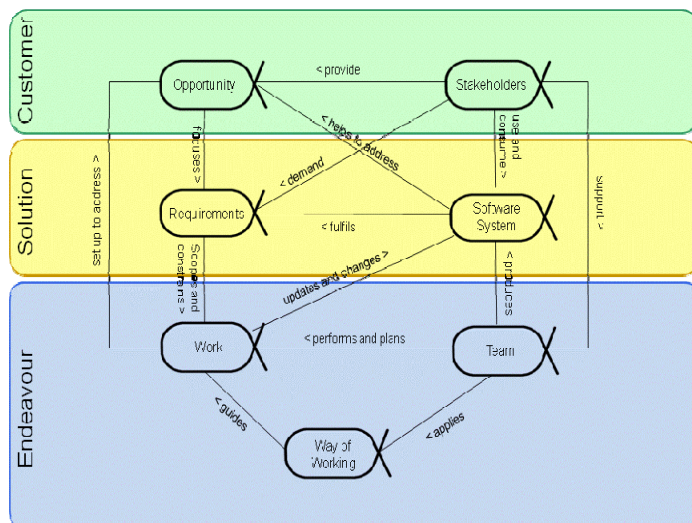


Figure 2A. Essence Alphas

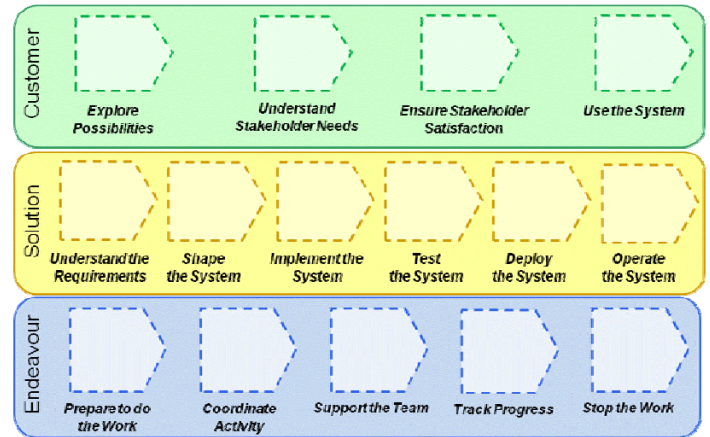


Figure 2B. Essence Activity Spaces



Figure 2C. Essence Competencies

Figure 2. Essence Kernel Elements

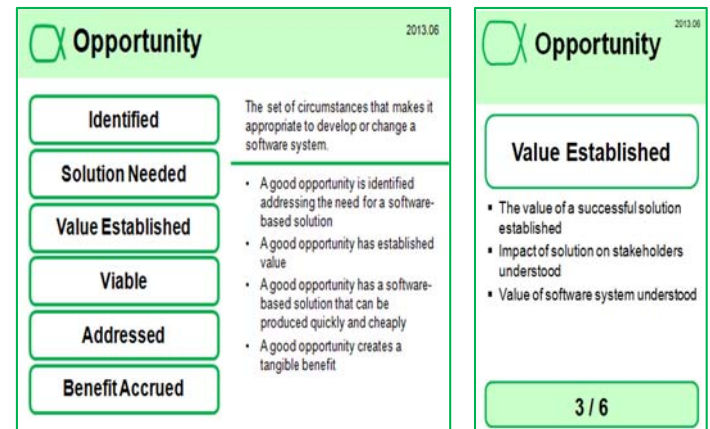


Figure 3. Essence Alpha State and State Checklist

3.2 Essence Activity Space and Goal State

Activity spaces are realized by one or more activities in a practice. Each activity may create or update one or more work products. For example, the *understand the requirements* activity space could be viewed as being realized by the release planning activity when using Scrum practice. Release planning produces a product backlog (work product), and this could be viewed as a manifest of the *requirements* alpha.

Activities within a given *activity space* in an *area of concern* changes the *states* of some *alphas* in the same *area*. Figure 4 shows, based on Essence 1.0 [3], the *alpha states* that should typically be achieved by each *activity space*. We will call them the *goal states of an activity space*. For instance, the goal states of the *explore possibilities* activity space include: *stakeholder::recognized*, *opportunity::identified*, *opportunity::solution needed*, and *opportunity::value established*. The goal *states of an activity* can be determined as a subset of the goal states of the activity spaces in which the activity is contained.

We can observe in Figure 4 that an *activity space* sometimes transforms an *alpha* to a new *state* over one or more intermediate *states*: e.g., the *understand the requirements* activity space transforms the *requirements* alpha from a null state to the *coherent* state through two intermediate states: *conceived* and *bounded*. When an *activity space* drives an *alpha* to go through multiple states, the *checklist* to verify achievement of the *activity space's* goal states should cumulatively contain all the *checkpoints* corresponding to those multiple *states*. The *activity space checklist* to verify achievement of the entire goal states of an *activity space* is obtained by the union of the *checklist* for every *alpha* affected by the *activity space*.

It should be noted that the *activity space-alpha state* relationships suggested in Essence 1.0 [3] (and shown in Figure 4) should be regarded as just a guideline. They will need to be tailored to address specific project situations. The Essence specification itself will evolve over coming years. The systematic procedure presented in Section 4 for describing a practice based on the Essence kernel is, however, still valid independent of any change to the *activity space-alpha state* relationships and *alpha state checklists*.

Alpha States Activity Spaces	Opportunity					Stakeholder				
	Identified	Solution Needed	Value Established	Viable	Addressed	Benefit Accrued	Recognized	Represented	Involved	In Agreement
Explore Possibilities										
Understand Stakeholder Needs										
Ensure Stakeholder Satisfaction										
Use the System										

4A. Customer Area

Alpha States Activity Spaces	Requirement					Software System				
	Conceived	Bounded	Coherent	Acceptable	Addressed	Fulfilled	Architecture Selected	Demonstrable	Usable	Ready
Understand the requirements										
Shape the System										
Implement the System										
Test the System										
Dploy the System										
Operate the System										

4B. Solution Area

Alpha States Activity Spaces	Team			Work				Way of Working			
	Seeded	Formed	Collaborating	Performing	Adjourning	Initiated	Prepared	Started	Under Control	Concluded	Closed
Prepare to Do the Work											
Coordinate Activity											
Support the Team											
Track Progress											
Stop the Work											

4C. Endeavor Area

Figure 4. Goal States of Activity Spaces

4. DESCRIPTION OF SCRUM PRACTICE USING ESSENCE KERNEL

Scrum and its variants are the most popular agile project management practices today. According to [20], 72% of about 4000 software practitioners surveyed used Scrum (54%), Scrum/XP hybrid (11%) or Scrumban (7%). There is a rich literature and web content on Scrum practice [21, 22, 23, 24]. We consider the following elements to comprise

the Scrum practice which will be translated into Essence kernel and language: Stakeholder, Product Envisioning, Product Vision, Release Planning, Product Backlog, Scrum Team, Product Owner, Scrum Master, Development Team, Sprint, Sprint Planning, Sprint Backlog, Task Board, Daily Scrum, Definition of Done, Work Remaining, Burndown Chart, Sprint Review, Sprint Retrospective and Product Increment. Definitions of those terms can be found in [21, 24].

Like other software engineering practices, Scrum practice has been textually described using a set of unique terminologies with a few graphical aids and tools (such as Task Board, Burndown Chart) to facilitate the understanding and enforcement of its ceremonies. The OMG Essence specification [3] includes an example that illustrates an Essence-based description of Scrum practice. The same Essence-based description of Scrum was compared with a SPEM 2.0-based description of Scrum in [17] to demonstrate the fundamental difference between Essence and SPEM 2.0. A key difference was found to be the notion of *alpha* with *alpha states* and the usage of *states* to help assess progress and provide planning guidance. However, neither [3] nor [17] explained how a practice can be systematically translated into an Essence-based description. To the best of the authors' knowledge this is the first time in literature that a robust procedure is presented for converting any software engineering practice into an Essence-based description.

Only *alphas* in the Essence kernel were used for evaluating the health of project progress in [4, 5]. Using only *alphas* was sufficient to evaluate the progress and health in a practice-agnostic way. However, to describe a practice using Essence kernel, we use all types of kernel elements—*alphas*, *activity spaces* and *competencies*—as well as *patterns* in this paper. We use all of them because existing practices are mostly defined in terms of activities (such as Sprint Planning), work products (such as Sprint Backlog) and roles (such as Development Team), which manifest *activity spaces*, *alphas* and *role patterns*, respectively. A *pattern* in the Essence language is a named structure made up of several Essence elements [3]. The *role pattern* associates practice activities, work products and competencies such that role R possesses competencies C to be able to perform activities A to produce work products W.

We apply the Activity-State Mapping Algorithm (Figure 5) presented in [25, 26] to assign practice activities to *activity spaces*, and to specify *alpha states* and *checklists* as the criteria to check the health and progress of each practice activity. By mapping practice activities to *activity spaces*, one can obtain the *goal states* for each practice activity based on Figure 4, and the Essence-provided *checklists* associated with each *goal state*. This supports the augmentation of existing practices with quality gates and governance procedures.

- Step 1:** Determine the set of activities to be performed in executing a chosen practice.

Step 2: Repeat for each activity listed in Step 1.

Step 2.1: Assign the activity to a set of *activity spaces*.

Step 2.2: Merge the *goal states* of the chosen set of activity spaces so as to generate the candidate goal states of the activity. (See Figure 4.) Determine the *goal states of the activity* as a subset of the candidate goal states.

Step 2.3: Collect all the *checkpoints* corresponding to the *goal states of the activity* listed in Step 2.2.

Step 2.4: Determine the *activity checklist* for the activity as a subset of the checkpoints collected in 2.3. (Note that the *activity checklist* is a subset of the union of *activity space checklists* for the activity spaces to which the activity was assigned.)

Figure 5. Activity-State Mapping Algorithm

4.1 Scrum Work Product to Alpha Mapping

Figure 6 shows an example of the mapping between the set of work products from Scrum practice and the set of Essence *alphas*.

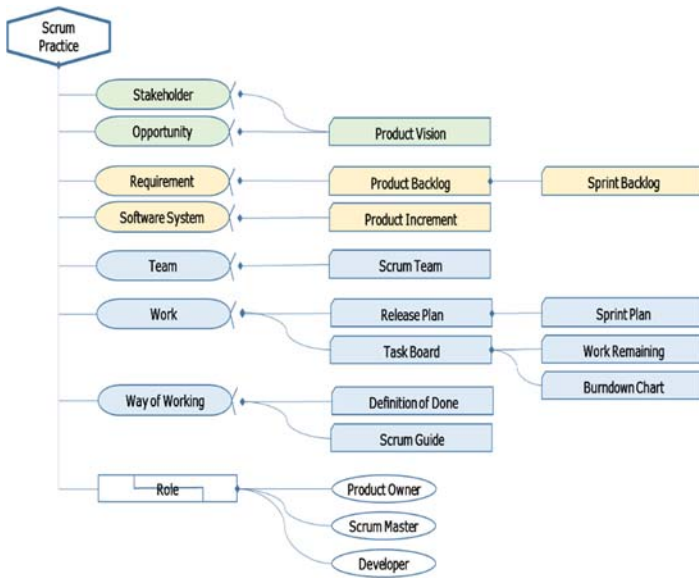


Figure 6. Mapping of Scrum Work Products to Essence Alphas

In this example, the Product Vision in Scrum practice manifests the *stakeholder* and *opportunity* alphas in Essence, which articulates the *stakeholders'* needs and the reason for creating the *software system*. The Product Backlog developed based on the Product Vision manifests the *requirements* alpha, which specifies what the *software system* should do to address the *opportunities* and satisfy the *stakeholders*. The Sprint Backlog is modeled as a *sub-alpha* of the Product Backlog. An *alpha* may contain a collection of other alphas. Together, these *sub-alphas* contribute to and drive the state of the *superordinate alpha* [3]. The Release Plan and Task Board are work products manifesting the *work* alpha, i.e., efforts put in order to build the *software system* successfully. The Sprint Plan, Work Remaining and Burndown Chart are modeled as *sub-alphas*. The definition of roles such as Product Owner, Scrum Master and Development Team is a major discriminant that sets Scrum practice apart from other practices. The roles of the Scrum Team members are defined using the *role pattern* as explained above.

It should be noted that Scrum work products relate to all 7 *alphas*. This means that Scrum practice is a well-balanced practice in so far as the work products address all important dimensions of the project. If it were not, for instance, for the Sprint Retrospective, the state of the *way of working* alpha might not progress from an Essence perspective.

4.2 Scrum Activity to Activity Space Mapping

Figure 7 shows the mapping of the set of activities in Scrum practice to the set of *activity spaces* in Essence kernel. Directed lines in Figure 7 that connect Scrum activities with each other are called *activity associations* in Essence [3]. *Activity association* represents an “end-before-start” association (viz., sequence flow).

Out of 15 *activity spaces* only 9 are addressed by Scrum practice. Unaddressed activity spaces are: *shape the system*, *implement the system*, *deploy the system*, *stop the work*, *use the system*, and *operate the system*. The partial coverage of *activity spaces* stems from the fact that Scrum practice focuses on the management of product and project, and does not provide guidance for development techniques such as requirement modeling, architecture design, coding and testing. Another limitation of Scrum practice is that it does not address the full life cycle of software delivery. It lacks guidance for release, transition and operation endeavors. Therefore, organizations often combine Scrum practice with other complementary practices such as agile modeling [27], agile

architecture design [28], extreme programming (XP) [29, 30], release and transition processes, asset reuse, etc. [28, 31] One of key advantages in mapping the practices to Essence kernel is the visibility it brings to potential gaps as we see in this section.

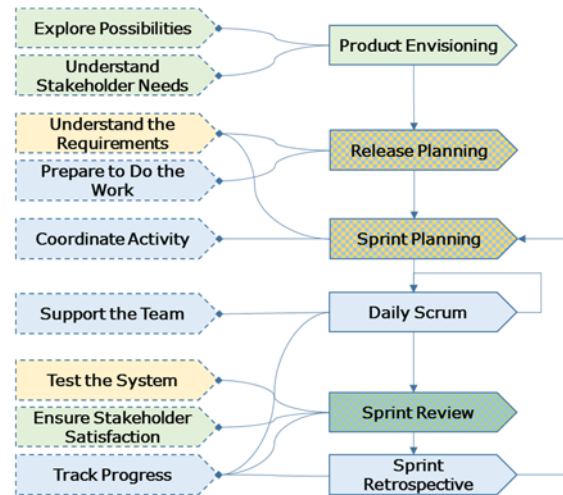


Figure 7. Mapping of Scrum Activities to Essence Activity Spaces

In Figure 7, the Product Envisioning activity is mapped to two *activity spaces*: *explore possibilities* and *understand stakeholder needs*. Likewise, the Release Planning and Sprint Review activities each realizes multiple activity spaces. On the other hand, multiple activities including Daily Scrum, Sprint Review and Sprint Retrospective are mapped to the *track progress* activity space. As such, the *activity-to-activity-space mapping* is many-to-many.

When an activity tries to realize multiple *activity spaces*, one may suspect that the given practice lacks a detailed guideline for that activity. Product Envisioning alone indeed requires a host of critical activities such as market analysis, competition analysis, key features selection, value proposition, financial projection, risk identification, etc. The Scrum practice does not go into details about those activities and the Product Owner will need to resort to other practices such as software product management (SPM) practices to deal with them. The same holds for Release Planning. The Product Owner may well want to combine some activities in the SPM discipline to deal with target market segmentation, internal and external positioning, non-functional requirements, product architecture, pricing, etc. On the other hand, Scrum requires several activities such as Daily Scrum, Sprint Review and Sprint Retrospective for the *track progress* activity space. This confirms that Scrum puts emphasis on project planning and tracking and provides guidelines at a detail level.

In Figure 8, different roles in Scrum practice are associated with Essence *competencies* (Figure 2C). As Scrum Teams are self-directed teams, Developers need to possess *leadership* and *management* competencies as well.

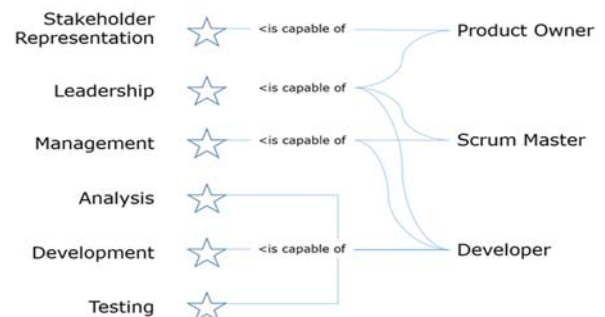


Figure 8. Mapping of Scrum Roles to Essence Competencies

It should be noted that there is not a single way to represent a practice, such as Scrum, in Essence. As an example, Sprint Planning could be mapped to the *prepare to do the work*, or the *coordinate activity*, or the *support the team* activity space, or any combination of them. The value of this exercise, regardless of how you choose to conduct this mapping, is the discussion that ensues within each organization related to potential gaps and subsequent practice improvement decisions.

4.3 Essence-Powered Scrum Diagram

Figure 9 summarizes an Essence-powered practice in what we call an Essence-Powered Practice Diagram. The lines with the “<produce>” label represent the relationship regarding which Scrum activities (Figure 7) produce which work products (Figure 6). The diagram also shows the assignment of activities to *activity spaces*. The abacus diagram on the right and the *activity space* column (third column) in Figure 9 reflect the

relationships between *activity spaces* and *alpha states* as described in Figure 4. For example, the *ensure stakeholder satisfaction* activity space should move the *stakeholder* and *opportunity* alphas to the states of *satisfied for deployment* and *addressed*, respectively.

By assigning each Scrum activity to appropriate *activity spaces*, one can obtain a comprehensive *checklist* (Figure 5) that can be used to evaluate if the activity is making healthy progress or has moved the project to a desirable target *state*. This is one of the real benefits that derives from describing Scrum using the Essence kernel.

It should be noted, however, that activities don’t necessarily stop when targeted *alpha states* are achieved; instead they may continue to ensure the team does not fall back. The project may even regress to former *states* as project conditions change.

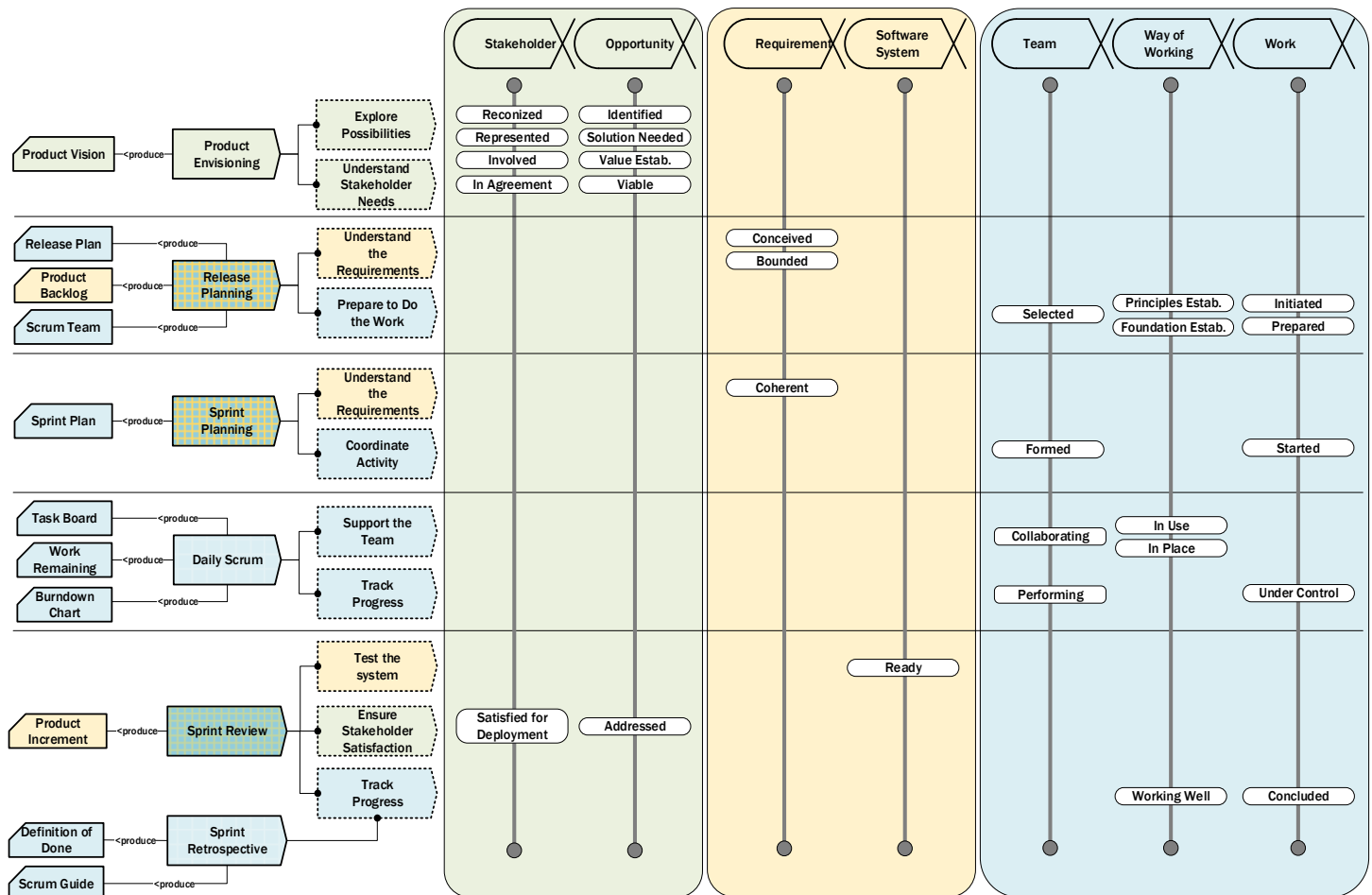


Figure 9. Essence-Powered Scrum Practice

4.4 Scrum Activity Definition Card and Activity Checklist

Figure 10 illustrates an *activity definition card* for the Release Planning activity. The card identifies the *initial states* and the *goal states* of the activity. The Release Planning activity may start when the *stakeholder* and *opportunity* alphas have reached the *in agreement* and *viable* states, respectively. When the activity fulfills its goal, the *requirements*, *team*, *way of working* and *work* alphas should reach the *coherent*, *selected*, *foundation established* and *prepared* states, respectively. The card also indicates that the Product Owner possessing the competency of *stakeholder representation* should conduct the Release Planning activity.

Figure 11 shows the checklist for the *coherent* state of the *requirements* alpha, as an example, which can be consulted while the Release Planning activity is in progress and when it is *done*. This checklist was compiled by aggregating the checkpoints for *Requirements::Conceived*, *Requirements::Bounded* and *Requirements::Coherent* states that are provided in Essence [3].

An *activity checklist* can be produced by aggregating the checklists for the *goal states* of the activity. For example, the checklist for the Release Planning activity is given by the collection of the checklists for *coherent* state of the *requirements* alpha, the *selected* state of the *team* alpha, the *foundation established* state of the *way of working* alpha and the *prepared* state of the *work* alpha, as shown in Figure 12.

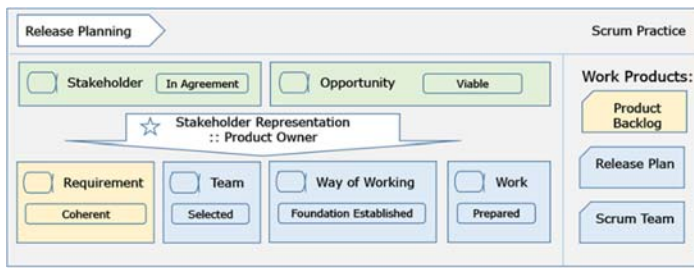


Figure 10. Activity Definition Card for Release Planning in Scrum

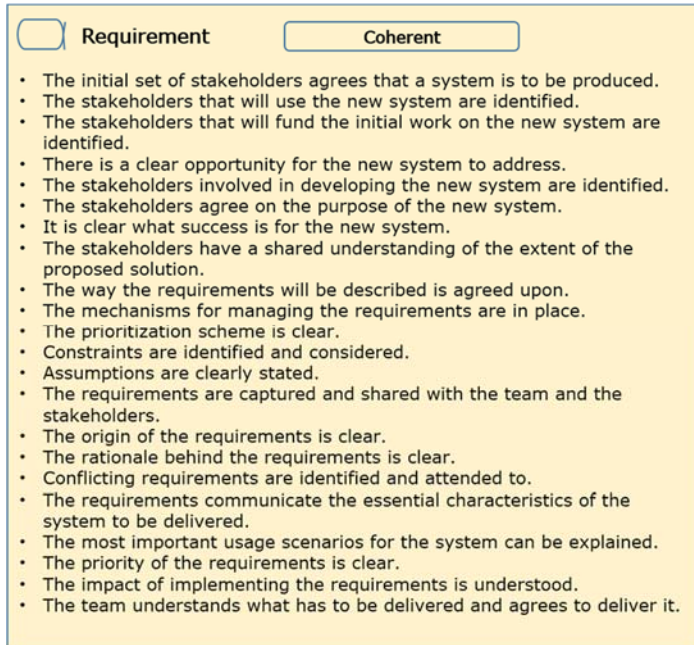


Figure 11. Alpha State Checklist



Figure 12. Activity Checklist

Other software engineering practices can be described using Essence kernel in the same manner as shown in this section for Scrum practice. In general, given the roles, activities and work products defined in a practice, we can associate the roles with Essence *competencies*, activities with *activity spaces*, and work products with *alphas*. We can then construct the Essence-powered practice as shown in Figure 9, where the *goal states* for each activity and its associated work products are specified. Based on Figure 9, we can generate the *activity definition card* for each activity performed in a practice as shown in Figure 10, together with the *activity checklist* as illustrated in Figure 12.

The *activity definition cards* provide concise reminders and cues for team members as they go about their daily tasks. By providing practical *checklists*, as opposed to conceptual discussions, the Essence-powered practice becomes something the team uses on a daily basis. This is a fundamental difference from traditional approaches, which tend to overemphasize method description as opposed to method use, and tend to be consulted only by people new to the team. The project method

assembled from Essence-powered practices becomes a small deck of cards in their pockets, which team members can easily pull out to discuss the current project state, work assignments and collaboration among them. Teams can also discuss areas of improvement by referring to the cards and their associated checklists. [5, 32]

5. CONCLUDING REMARKS

The motivation behind using Essence comes from the pain points shared by software engineering practitioners today.

- A large number of software engineering practices (such as use case analysis, test-driven development, scrum project management to name a few) are available, but each in different notation.
- Learning a new software engineering practice is difficult even if one already knows other similar practices.
- Practices do not provide developer-friendly forms of guidelines and checklists so as to be consulted in daily activity planning and tracking.
- In the absence of multidimensional checklists that ensure the health of a project from all aspects throughout its lifecycle, developers often overlook some of the aspects, ending up with a software system not delivering promised values to the stakeholders.
- It is difficult to compare, select and integrate different practices into a streamlined method for a software engineering project.
- It is also difficult to build an enterprise method architecture in a company that requires a variety of standardized methods and practices for software engineering for different product lines and service lines.

Benefits that derive from describing Scrum practice (or any other practice) using Essence kernel are manifold:

- Using Essence-provided *alpha state checklists* and *activity checklists*, the Scrum Team can evaluate their progress and early detect systemic problems (such as increasing technical debt, loss of stakeholder support, dysfunctional team work, etc.) [4, 5, 7, 8, 9, 32]. The key measure of progress of the native Scrum practice is the amount of working software the team produces and the speed with which the team produces it. Essence-provided *alpha state checklists* and *activity checklists* complement these measures by providing other views of the progress and health of the project [10, 12, 15, 16].
- The mapping of Scrum activities and work products to Essence kernel elements allows the Scrum Team to find deficiencies in the breadth and depth of the Scrum practice in use [6, 26, 33]. For instance, the team may add some agile modeling practice [27] to better clarify Product Backlog Items during the Release Planning for a large and complex problem domain; add some agile architecture practice [28] to satisfy the *architecture selected* state of the *software system alpha*; or add extreme programming practices [29] to better enforce the *implement the system* and *test the system* activity spaces [16, 30].
- When the team adds other practices to the Scrum practice, it can first describe all the selected practices using Essence kernel and then systematically assemble them into a coherent project method [26, 33]. Describing each practice using the common Essence kernel allows the team to find gaps, overlaps, conflicts and complements among selected practices so that the team may integrate them into a well-formed method [10, 12, 15, 16].
- If the Scrum Team carefully adjusts the *activity-to-activity-space mapping* taking into consideration the team's competency and learning curve. For example, if the team is new to Scrum, it may map the Sprint Planning activity to the *coordinate activity* activity space in initial sprints, and later to the *support the team* activity space, as it gains more experience and better capability [33]. The *support the team* activity space requires the project to reach more challenging

states of the *team*, *work* and *way of working* alphas, thereby improving the team productivity (see Figure 4C).

- Different teams in an organization may apply different variations of Scrum mixing different sets of other practices such as agile modeling, XP, Kanban, Unified Process, etc. By describing all the practices and methods using the common Essence kernel, the organization can establish and maintain an enterprise method architecture [34]. An Essence-based enterprise method architecture helps the organization find common practices across teams, lets them share and collaboratively improve common practices, learn other teams' new successful practices, train new employees relevant practices depending on their roles, and enable employees to quickly adapt to their new team's specific method [10, 12, 15, 16]. The organization thus becomes a truly learning organization.
- It becomes easier to manage a large project consisting of multiple teams using different sets of practices. Such a large project can consolidate different practices into a single streamlined method for the entire project and check its progress using the common Essence kernel and checklists. Furthermore, an organization with an Essence-based method architecture can monitor and control, perhaps with an integrated dashboard, the health and progress of every ongoing project using the common set of *alpha state checklists* [10]. Essence kernel provides a structure and mechanism for progress monitoring, retrospectives, risk management and project steering in a holistic, simple, lightweight, non-prescriptive and method-agnostic fashion [18].

6. ACKNOWLEDGMENT AND CAVEAT

This work was supported by ICT R&D program of MSIP/IITP in Korea [14-824-10-003: Development of Cloud Services for Software Engineering Method Enactment Based on the OMG Essence Standard]. The authors thank Pekka Abrahamsson, Ivar Jacobson, Svante Lidman and Roly Stimson, and an anonymous associate editor of ACM SIGSOFT SEN for helpful comments that improved the paper.

The authors are members of SEMAT and some of them are members of the OMG Essence Revision Task Force. However, the content of this paper does not represent the official views or policies of SEMAT (Software Engineering Method and Theory) nor of OMG (Object Management Group).

7. REFERENCES

- [1] Kennaley, M. 2010. *SDLC 3.0: Beyond a Tacit Understanding of Agile*, Fourth Medium Press.
- [2] Park, J. S., Jacobson, I., Myburgh, B., Johnson, P., and McMahon, P. E. 2014. *SEMAT Yesterday, Today and Tomorrow*. SEMAT. DOI=<http://semat.org/wp-content/uploads/2014/12/SEMAT-Yesterday-Today-and-Tomorrow-v1.0.pdf>.
- [3] Object Management Group, 2014. *Essence—Kernel and Language for Software Engineering Methods 1.0*. DOI=<http://www.omg.org/spec/Essence/>.
- [4] Jacobson, I., Ng, P. W., McMahon, P. E., Spence, I., and Lidman, S. 2012. The essence of software engineering: the SEMAT kernel. *Communications of the ACM* 55 (Dec. 2012), 42-29.
- [5] Jacobson, I., Ng, P. W., McMahon, P. E., Spence, I., and Lidman, S. 2013. *The Essence of Software Engineering: Applying the SEMAT Kernel*, Addison-Wesley.
- [6] Park, J. S. 2015. Software engineering in the context of business systems—how Essence can help. In *Software Engineering in the Systems Context*, I. Jacobson and H. Lawson, Ed. College Publications, London.
- [7] Jacobson, I., Spence, I., and Ng, P. W. 2013. Agile and SEMAT—perfect partners. *Communications of the ACM* 56 (Nov. 2013) 53-59.
- [8] Jacobson, I., Ng, P. W., Spence, I., and McMahon, P. E. 2014. Major-league SEMAT—why should an executive care? *Communications of the ACM* 57 (April 2014) 44-50.
- [9] Jacobson, I. and Seidewitz, E. 2014. A new software engineering. *Communications of the ACM* 57 (Dec. 2014) 36-41.
- [10] Seymour, E. 2015. We all have different ways to do things and that's OK. In *Essence-in-Practice Conference* (Berlin, June 18, 2015). OMG Technical Meeting. DOI=<http://www.omg.org/news/meetings/tc/berlin-15/special-events/essence-presentations/seymour.pdf>.
- [11] Cunningham, D. 2013. Enabling Fujitsu's industrialized delivery of application services. In *Essence Information Day* (Berlin, June 20, 2013). OMG Technical Meeting. DOI=<http://www.omg.org/news/meetings/tc/berlin-13/special-events/essence-pdfs/S7-Cunningham.pdf>.
- [12] Nadin, S. 2015. Using Essence to deliver together: practical experience at Fujitsu. In *Essence-in-Practice Conference* (Berlin, June 18, 2015). OMG Technical Meeting. DOI=<http://www.omg.org/news/meetings/tc/berlin-15/special-events/essence-presentations/nadin.pdf>.
- [13] Perkins-Golomb, B. 2013. Applying SEMAT concepts at Munich Re: personal reflections. In *Essence Information Day* (Berlin, June 20, 2013). OMG Technical Meeting. DOI=http://www.omg.org/news/meetings/tc/berlin-13/special-events/essence-pdfs/S6_Burkhard.pdf.
- [14] McDonough, A. 2014. Munich Re and Essence—kernel and language for software engineering methods: a case study. OMG. DOI=http://www.omg.org/news/whitepapers/Munich_Re_Essence_Case_Study-2014-12-01_JP.pdf.
- [15] Perkins-Golomb, B. 2015. Successful utilization of ESSENCE at Munich Re. In *Essence-in-Practice Conference* (Berlin, June 18, 2015). OMG Technical Meeting. DOI=<http://www.omg.org/news/meetings/tc/berlin-15/special-events/essence-presentations/perkins-golomb.pdf>.
- [16] Ivar Jacobson International, 2014. Asian telecommunications equipment vendor successfully achieves rapid and sustainable agile transformation. DOI=http://www.ivarjacobson.com/uploadedFiles/Pages/Knowledge_Centre/Resources/Case_Studies/Resources/AsianTelecomm1.pdf.
- [17] Elvesæter, B., Benguria, G., and Ilieva, S. 2013. A comparison of the Essence 1.0 and SPEM 2.0 specifications for software engineering methods. In *Proceedings of the 3rd Workshop on Process-Based Approaches to Model-Driven Engineering* (Montpellier, France, July 2, 2013). DOI=<http://dl.acm.org/citation.cfm?id=2489835>.
- [18] Péraire, C. and Sedano, T. 2014. State-based monitoring and goal-driven project steering: field study of the SEMAT Essence framework. In *Proceedings of the 36th International Conference on Software Engineering* (Hyderabad, India, May 31-June 7, 2014). DOI=<http://dl.acm.org/citation.cfm?id=2591155>.
- [19] Zapata, C. and Jacobson, I. 2014. A first course in software engineering method and theory. *DYNA* 81 (Jan./Feb. 2014) National University of Columbia. DOI=http://www.scielo.org/co/scielo.php?pid=S0012-73532014000100026&script=sci_arttext.
- [20] VersionOne, 2012. *7th Annual State of Agile Development Survey*. DOI=<https://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf>.
- [21] Schwaber, K. 2004. *Agile Project Management with Scrum*, Microsoft Press, Redmond.
- [22] Scrum.org. 2015. DOI=<https://www.scrum.org/>.
- [23] Scrum Alliance. 2015. DOI=<http://www.scrumalliance.org/>.

- [24] Schwaber, K. and Sutherland, J. 2015. *Scrum Guide*. DOI=<https://www.scrum.org/Scrum-Guide>.
- [25] Park, J. S. 2015. Essence-based goal-driven adaptive software engineering. In *Proceedings of the IEEE/ACM 4th SEMAT Workshop on General Theory of Software Engineering (GTSE)* (Florence, Italy, May 18, 2015). DOI=<http://dl.acm.org/citation.cfm?id=2820176>.
- [26] Park, J. S. 2015. Essence-powered Scrum: a generic approach to describing practices using Essence kernel and language. In *Essence-in-Practice Conference* (Berlin, June 18, 2015). OMG Technical Meeting. DOI=<http://www.omg.org/news/meetings/tc/berlin-15/special-events/essence-presentations/park.pdf>.
- [27] Ambler, S. 2015. *Agile Modeling—Effective Practices for Modeling and Documentation*. DOI=<http://www.agilemodeling.com/>.
- [28] Leffingwell, D. 2015. *Scaled Agile Framework*. DOI=<http://scaledagileframework.com/>.
- [29] Beck, K. and Andres, C. 2005. *Extreme Programming Explained*, Addison-Wesley, Boston.
- [30] Williams, L., Brown, G., Meltzer, A., and Nagappan, N. 2011. Scrum + engineering practices: experiences of three Microsoft teams. In *Proceedings of the 2011 International Symposium on Empirical Software Engineering and Measurement (ESEM)*. (Banff, Canada, Sept. 22-23, 2011). DOI=<http://dl.acm.org/citation.cfm?id=2083397>.
- [31] Ambler, S. and Lines, M. 2011. *Disciplined Agile Delivery: An Introduction*, IBM Software, Somers, NY. DOI=<http://www.disciplinedagiledelivery.com/>.
- [32] McMahon, P. E. 2014. *15 Fundamentals for Higher Performance in Software Development*, Leanpub.
- [33] Park, J. S. 2014. Essence-based adaptive software engineering. In *the 4th International Conference on Emerging Applications of Information Technology (EAIT)*. (Kolkata, India, Dec. 19, 2014). Computer Society of India. DOI=<https://sites.google.com/site/csieait/home>; <http://www.scribd.com/doc/251364248/Essence-Tutorial-JP>.
- [34] Park, J. S. 2013. Essence kernel-based enterprise method architecture. In *Essence Information Day* (Berlin, June 20, 2013). OMG Technical Meeting. DOI=<http://www.omg.org/news/meetings/tc/berlin-13/special-events/essence-pdfs/S5-Park.pdf>.