Modeling Software Product Line Engineering with Essence Framework

Eray TÜZÜN[1], Görkem GİRAY[2], Bedir TEKİNERDOGAN[3], Yagup MACİT[4]

[1]Department of Computer Engineering, Bilkent University, Ankara, Turkey
[2]Independent Researcher, İzmir, Turkey
[3] Information Technology Group, Wageningen University, Wageningen, The Netherlands
[4] Information and Communication Technologies Department, Havelsan, Ankara, Turkey
eraytuzun@cs.bilkent.edu.tr, gorkemgiray@gmail.com, bedir.tekinerdogan@wur.nl, ymacit@havelsan.com.tr

(Geliş/Received:13.11.2017; Kabul/Accepted:25.01.2018)
DOI: 10.17671/gazibtd.351731

***Abstract*—** Although several software product line engineering (SPLE) methods have been described in the literature, adopting these methods in practice is often not straightforward. Thorough understanding of the methods and their artefacts is necessary to apply the methods in a proper manner, and likewise realize the expected goals of SPLE. Recently the Essence framework has been proposed to model the essential elements of a method and to support the modeling of a broad set of software development methods including plan-driven methods and agile methods. So far, the Essence framework has been applied to single system development methods and not yet for SPLE methods. To enhance the understanding of SPLE methods and support a vision for tailoring SPLE methods, we provide a mapping of an SPLE method to the Essence framework. We present experiences about modeling an SPLE method using the Essence framework within the industrial context of Havelsan.

***Keywords*—** Essence Framework, Software Product Line Engineering (SPLE), Process modelling, Software Product Lines, Software process

Yazılım Ürün Hattı Mühendisliği Sürecinin Essence Çerçevesi ile Modellenmesi

***Özet*—** Literatürde birçok yazılım ürün hattı mühendisliği (YÜHM) yöntemi tarif edilmiş olmasına rağmen bu yöntemlerin pratikte uygulanması çoğu zaman kolay olmamaktadır. YÜHM'in beklenen hedeflerini gerçekleştirmek ve yöntemleri doğru uygulamak için hem yöntemleri hem de yapısal öğeleri tam olarak anlamak gereklidir. Essence çerçevesi, son zamanlarda plan güdümlü yöntemler ve çevik yöntemler de dahil olmak üzere geniş bir alana yayılmış yazılım geliştirme yöntem setinin modellemesi için önerilmektedir. Şu ana kadar, sadece tekil sistem geliştirme yöntemine uygulanan Essence çerçevesi henüz YÜHM yöntemleri için uygulanmamıştır. Bu çalışmada, YÜHM yöntemlerinin daha iyi anlaşılmasını sağlamak ve uyarlaması konusunda bir vizyon oluşturmak için seçilen YÜHM yönteminin Essence çerçevesi ile modellenmesi gerçekleştirilmiştir. Çalışma sonucunda, Essence çerçevesi ve YÜHM hakkında elde edilmiş olan deneyim ve öğrenilmiş dersler, endüstriyel olarak Havelsan bağlamında, sunulmuştur.

***Anahtar Kelimeler*—** Essence Çerçevesi, Yazılım Ürün Hattı Mühendisliği, Yazılım Ürün Hatları, Süreç modelleme, Yazılım süreçi

1. INTRODUCTION

The main motivation for transitioning to SPLE is to develop products more efficiently, to produce them with higher quality and get them to market faster [23][22]. To this end, different software product line engineering methods have been proposed such as, Pohl et al. [22], the Philips' CoPAM method [2], the SEI's Framework for Software Product Line Practice [6], the Fraunhofer's PULSE-approach [3], the FAST approach [27], and the Gomaa's PLUS approach [11].

Each of these methods have been defined at different precision levels. Some of these methods have been defined in a more generic and abstract manner while others provide a more thorough description of the artefacts, the rules and the relationships among these. In practice, we can observe that the application of these methods is not always straightforward and requires a thorough analysis of the method and a description of the method artefacts. Besides of understanding the SPLE method, it is important to track and monitor the process activities and likewise manage the overall project.

Thorough understanding of the methods and their artefacts is necessary to tailor and apply the method in a proper manner, and likewise support the achievement of the expected goals. Method engineering aims to design, construct and tailor methods, techniques and tools for engineering systems [4]. Situational method engineering describes the construction and tailoring of a system (or software) development method from atomic method building blocks (named as practices in Essence specification), conformant to the conceptual definitions in an underpinning metamodel (Essence Language in Essence specification) [13]. The construction and/or tailoring should be made according to some criteria (such as presented in [5] and [15]) specific to the attributes of the project at hand.

In this study, we report on our experiences within the industrial context of Havelsan, on applying the Essence framework to model the SPLE process and to support the tracking and monitoring of the SPLE process activities. Havelsan [12] is a large-scale systems and software company, which delivers products in the domain of simulation systems, command and control systems, and e-government applications. Havelsan has grown out of a history of delivering large, independent and custom developed solutions for specific requirement sets for their customers. In the current state, similar products are developed using single system development approach. Within the company, it has been decided to invest more in systematic software reuse and apply SPLE practices in potential projects. There are several potential candidates for applying SPLE approach in the company. For this study, we focus on a potential product line in the image processing domain. Although the company had extensive experience in the image processing domain, they had limited information on SPLE and practical application. In our earlier study, we have applied DSS (Decision Support System) to assess the feasibility of the image processing product line [26] and decided to apply SPLE.

To apply the SPLE process, it is important that the process is well-defined. So far, the Essence framework has until now only been applied to single system development methods and not yet for SPLE methods. To enhance the understanding of SPLE methods and prepare it for the application within Havelsan, we provide a mapping of an SPLE method to the elements of the Essence framework. We also provide an approach for tracking the progress of the application of an SPLE project within Havelsan. In sum, the research questions that we focus in this paper can be formulated as follows:

- *RQ1: How to apply the Essence framework to model an SPLE process?*
- *RQ2: What are the lessons learned with respect to modeling a two-life cycle process with the Essence framework?*
- *RQ3. What are the identified limitations of the SPLE process after modeling it with the Essence Framework?*

In Section 2, background information on the Essence Framework and SPLE are provided. In Section 3, we describe our approach for mapping SPLE to the Essence framework and present the results of this mapping in Section 4 in detail. In Section 5, we discuss the benefits and the lessons learned of our approach. In Section 6, related work is presented and finally, Section 7 concludes the paper.

2. BACKGROUND

2.1 Essence Framework

The Essence framework provides a common language and domain model of software development, which form a basis for modeling development methods. With the use of the Essence framework, development methods can be better compherended, learned and stack up against with other methods. The Essence framework can be used for modeling a wide set of software development methods [18] [14], including plan-driven methods such as Unified Process, and agile methods such as Scrum [20]. It was published by Object Management Group (OMG) and built by the work of the Software Engineering Method and Theory (SEMAT) community [25]. The Essence framework also proposes a way to track the overall state of a project and select activities for progress ending up with a concrete guidance for software development teams.

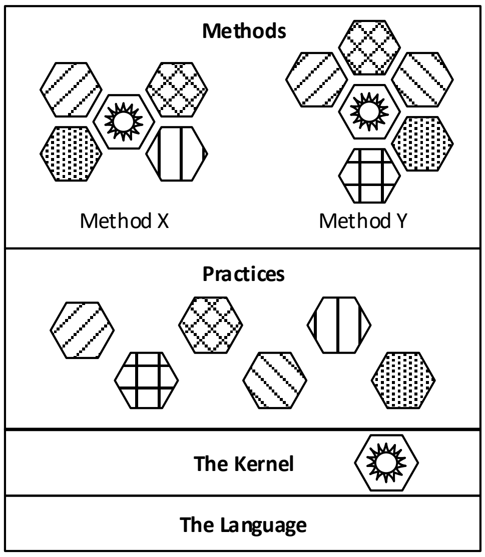


Figure 1. Layered architecture of Essence [21]

Alphas represent important dimensions in software development. Essence framework defines 7 Alphas defined in Essence Kernel. These Alphas are also organized into 3 areas of concern, namely Customer, Solution and Endeavor. The Customer area of concern consists of Opportunity and Stakeholders alphas. The Solution area of concern involves Requirements and Software System alphas. The Endeavor area of concern consists of Team, Work and Way of Working alphas.

Activity spaces are abstract containers for concrete activities. Essence Kernel defines activity spaces and forms a basis for practices to define concrete activities to achieve the goals of these activity spaces in terms of sets of Alpha states.

Key capabilities to perform activities are defined as competencies. Competencies (such as Development) are abstract and general containers for specific skills (such as Java programming) required during software development. Essence Kernel only covers general competencies as it aims to form a base for all software development methods.

2.2 Software Product Line Engineering

In this study, we use the SPLE approach as defined by Pohl et al. [22] as shown in Figure 2. Here, the domain engineering is responsible for defining the commonality and the variability of the product line whereas application engineering is responsible for deriving product line applications from the platform established in domain engineering.

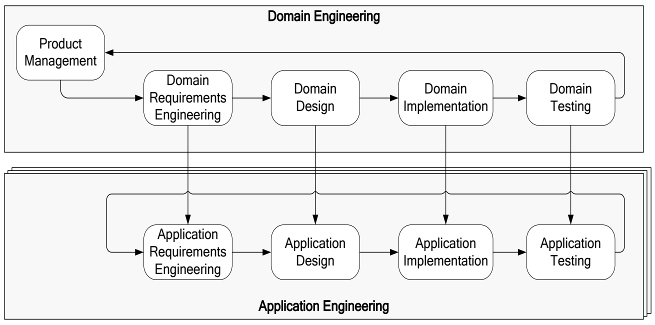


Figure 2. General SPLE Process [22]

In general, there appears to be a consensus that the SPLE process consists of life cycle processes of domain engineering and application engineering. This common SPLE process is shown in Figure 2. The domain engineering process is responsible for establishing the reusable platform and thus for defining the commonality and the variability of the product line. The platform consists of all types of software artefacts (requirements, design, realization, tests, etc.). The domain engineering process is composed of five key sub-processes: product management, domain requirements engineering, domain design, domain realization, and domain testing. In the application engineering process, the applications of the product line are built by reusing the artefacts and exploiting the product line variability as defined in the domain engineering process. The application engineering process is composed of the sub-processes application requirements engineering, application design, application realization, and application testing

3. SYSTEMATIC APPROACH FOR MAPPING SPLE to ESSENCE FRAMEWORK

For modeling methods to the Essence framework, a general approach was proposed in [10]. In this paper, we adapt this approach and apply it for modeling the SPLE method in [22] using the Essence framework. Our approach encompasses 6 steps, which are typically performed iteratively and incrementally. The description of each step is as follows:

Step 1. Extract concepts from SPLE method specification

After selecting the SPLE method, a domain analysis [16] applied to extract the key concepts of the method. Domain analysis focuses on identifying the common and variant elements of a domain, in this case the selected method. The domain model is defined as a glossary that includes the identified concepts. For the given SPLE method, we extracted the important concepts such as Domain Requirements, Application Requirements, and Application Requirements Specification.

Step 2. Classify extracted SPLE concepts according to Essence concepts

The identified SPLE concepts are analyzed and classified with respect to the categories as defined in the Essence framework. In the mapping process, different possibilities can occur. First, an SPLE element could be directly mapped to an Essence framework element. For example, in Table 1, Application Requirements Specification is a work product and could be mapped to the Essence Framework. In other cases, the SPLE element might not be explicitly in the Essence framework category. In that case, an extension can be defined in the Essence framework. For example, Domain Requirements can be modelled as a sub-alpha of requirement by extending the Essence kernel.

Table 1. Example classification of SPLE concepts based on Essence framework

SPLE Concept	**Essence Concept**	**Mapping Type (Direct, Extension)**
Domain Requirements	Requirements Sub-Alpha	Extension
Application Requirements	Requirements Sub-Alpha	Extension
Domain Requirements Elicitation	Activity	Direct
Application Requirements Specification	WorkProduct	Direct

The mapping process can also provide insight into the areas that are not directly addressed by the SPLE method. That is, after the mapping process we might identify that several Essence framework elements have not been considered. For example, *Deploy the System* activity space is not directly addressed by the SPLE method. This situation can trigger the need to complement the missing elements in the SPLE method with elements of other methods. A partial result of the first two steps are shown in Table 1.

Step 3. Define SPLE concepts using Essence Language

SPLE concepts are specified using graphical language elements provided by the Essence framework. As an example, the definition of Requirements alpha in Essence language is shown in Figure 3.

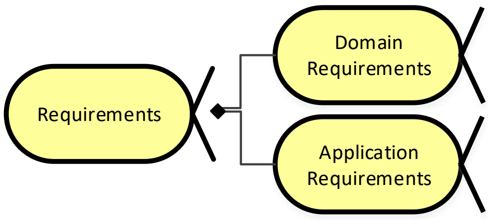


Figure 3. Sub-alphas of Stakeholders alpha in Essence framework

Step 4. Define properties of SPLE concepts

In this step, the properties of the mapped concepts are specified. For instance, we have described the level of details in *Application Requirements Specification* work product. Level of detail is defined as the specification of the amount of detail or range of content in a work product in the Essence framework. The level of detail properties of *Application Requirements Specification* work product is shown in Figure 4.

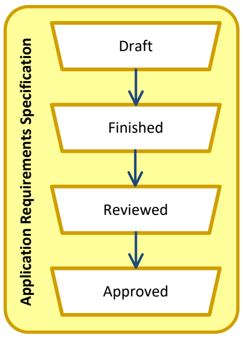


Figure 4. Level of Detail for Application Requirements Specification work product

Here, the initial level of detail of the Application Requirements Specification is in *draft* mode, once the document is completed, the level of detail becomes *Finished*. With the review of the document by the internal stakeholders, the level of details becomes *Reviewed*, and finally with the review of the document by the external stakeholders, the level of detail becomes *Approved*.

Step 5. Associate related concepts

In this step, the associations among the related concepts are demonstrated. After specifying properties of concepts (Step 4) and associating related concepts, cards can be prepared. Cards are used as visualization elements to summarize the essential properties of an element in the kernel language.

Step 6. Review the process

In this step, the mapped method is reviewed for its completeness and quality.

4. MAPPING SPLE CONCEPTS TO ESSENCE FRAMEWORK

In this section, we deliver the results of the mapping of the SPLE method as defined by Pohl et al. [22] to the Essence Framework. This mapping is done with key members from the SPLE project team and process engineers in the company with the guidance of the authors. The Essence framework provides Practice Workbench (https://www.ivarjacobson.com/esswork-practice-workbench), which provides a development environment for composing, authoring, collaborating and managing software development practices and methods. In this study, we used this Practice Workbench to author and tailor our own SPLE methodology. Hereby, we use the approach as defined in section 3. From section 4.1 through section 4.3, we discuss the mappings of alphas, activities, and competencies to the Essence framework. In section 4.4, we present a usage scenario to track the SPLE progress using the Essence framework.

4.1 Mapping of SPLE Sub-Alphas and Work Products to Essence Alphas

Originally, the Essence framework defines 7 alphas to track the progress and health of a project. The Essence framework has been so far mainly applied to single system development which closely maps to the 7 alphas in the Essence framework.

The selected SPLE method, as summarized in Section 3, defines a two-life cycle process, application engineering and domain engineering. This distinction between two lifecycle processes has a direct impact on defining the sub-alphas. In principle, this means that we need two different types of sub-alpha for modeling both application engineering and domain engineering elements.

We have shown this in Figure 5. For stakeholders, we have defined separate sub-alphas for *Product Line Stakeholders*, *Application Stakeholders* and *Domain Stakeholders*. Here, *product line stakeholders* concern with the managerial aspects of the product line. *Domain Stakeholders* concerns with the domain in general whereas *Application Stakeholders* only concerns with the specific application. In addition, we should note that we might have different stakeholders for each application in the product line. In a similar manner, we have distinct alphas for *Domain Opportunity* and *Application Opportunity*. The *Domain Opportunity* defines the set of circumstances that makes it suitable to develop or change the whole product line, whereas *Application Opportunity* is related with the individual application.

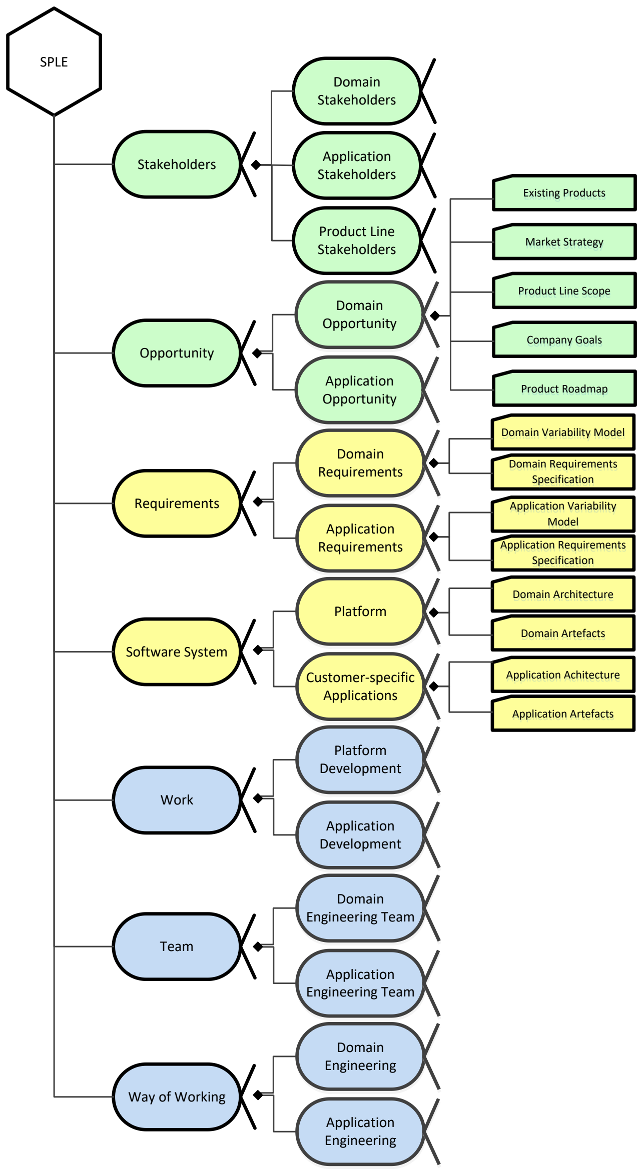


Figure 5. Mapping of SPLE sub-alphas and work products to Essence alphas

After defining the necessary alphas, it is also important to map the *work products* to the *alphas*. Work Products in the Essence Framework represent the concrete things to work with, providing evidence for the *alpha* states. For example, for *domain opportunity* alpha we have identified market strategy, product line scope, company goals, and product roadmap as work products. These work products get updated by the related activities that we will define in the next subsection

4.2 Mapping of SPLE Activities to Essence Activity Spaces

The Essence framework defines *Activity Spaces* for each three areas of concern (Customer, Solution and Endeavor). *Activity spaces* define essential things to do in a software project. The mappings from activity spaces to activities can be many-to-many. This mapping activity is crucial since it gives an overview of which activity spaces addressed and which activity spaces are not addressed by a specific method. For SPLE, we have mapped the activities that are described in Pohl et al. [22] to activity spaces in the Essence framework in Figure 6. If there are any *Activity Spaces* that are not addressed by the activities of SPLE, this means these *Activity Spaces* should be addressed by other means, for example using elements of other software development methods.

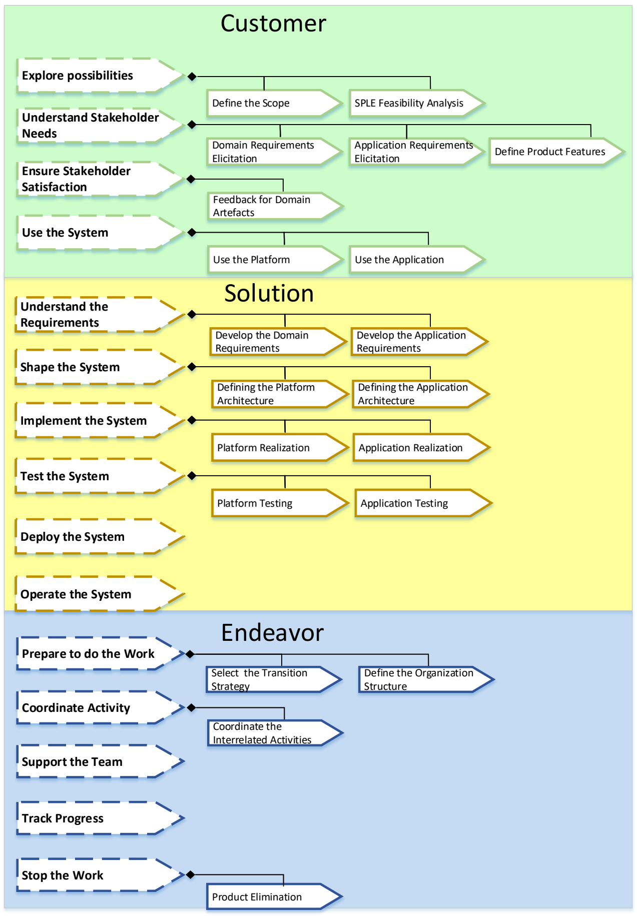

Customer
Explore possibilities
Define the Scope
SPLE Feasibility Analysis
Understand Stakeholder Needs
Domain Requirements Elicitation
Application Requirements Elicitation
Define Product Features
Ensure Stakeholder Satisfaction
Feedback for Domain Artefacts
Use the System
Use the Platform
Use the Application
Solution
Understand the Requirements
Develop the Domain Requirements
Develop the Application Requirements
Shape the System
Defining the Platform Architecture
Defining the Application Architecture
Implement the System
Platform Realization
Application Realization
Test the System
Platform Testing
Application Testing
Deploy the System
Operate the System
Endeavor
Prepare to do the Work
Select the Transition Strategy
Define the Organization Structure
Coordinate Activity
Coordinate the Interrelated Activities
Support the Team
Track Progress
Stop the Work
Product Elimination

Figure 6. Mapping of SPLE activities to Essence Activity

As illustrated in Figure 6, the customer area of concern includes the activity spaces; *Explore Possibilities*, *Understand the Stakeholder Needs*, *Ensure Stakeholder Satisfaction*, and *Use the System*. In the *Explore the Possibilities*, the product line opportunity to be addressed is analyzed and the stakeholders are identified. This activity space is addressed by *SPLE Feasibility Analysis* and *Define the Scope* activities in SPLE.

Understand the Stakeholder needs is addressed by *Domain Requirements Elicitation*, *Application Requirements Elicitation* and *Define Product Features* activities. In the *Ensure the Stakeholders* activity space, SPLE defines an activity for *Feedback for Domain Artefacts*. This activity defines the feedback mechanism from application engineering to domain engineering. Finally, for *Use the Systems* activity space, SPLE defines *Use the Platform* and *Use the Application* activities.

The solution area of concern includes the activity spaces; *Understand the Requirements*, *Shape the System*, *Implement the System*, *Test the System*, *Deploy the System* and *Operate the System*. For *Understand the Requirements*, the SPLE method defines two main activities *Develop the Application Requirements* and *Develop the Domain Requirements*. We can observe from Figure 6 that *Understand the Requirements*, *Shape the System*, *Implement the System* and *Test the System* activity spaces are addressed. However, *Deploy the System* and *Operate the System* activity spaces are not addressed. The SPLE method leaves it to the users of the method to define their own activities and practices. Here DevOps practices can be integrated into the method to address these activity spaces.

For *Endeavor* area of concern, the activity spaces are *Prepare to do the work*, *Coordinate Activity*, *Support the Team*, *Track Progress* and *Stop the Work*. *Prepare to do the Work* activity space is addressed by *Select the Transition Strategy* and *Define the Organization Structure* activities. Coordination among domain engineering and application engineering is addressed by the *Coordinate the Interrelated Activities*. The SPLE method purposely does not define a life cycle process but rather leave it to the users of the method to decide on a lifecycle method. To use SPLE method, this endeavor area of concern needs to be supported by defining a lifecycle approach and practices. These activity spaces could be addressed by for example adding Scrum practices [24]. Scrum might help addressing the *Support the Team* activity space by sprint retrospective, and *Track Progress* activity space by daily scrum and sprint review activities.

Each of these activities can be defined in detail using the *Activity Definition Cards*. Example activity definition cards are illustrated in Appendix A and Appendix B, for *Develop the Application Requirements* and *Develop the Domain Requirements* activities respectively

4.3 Mapping of SPLE Roles to Essence Competencies

In the Essence framework, competency is defined as a set of the capabilities, attainments, knowledge, and skills necessary to do a certain kind of work [18]. The Essence framework defines 6 key competencies as it was discussed in Section 2.2.

The SPLE process defines several roles for Application Engineering and Domain Engineering. In Figure 7, we have mapped these roles to the competencies that are defined in the Essence framework. Having a competency mapping to SPLE role would be useful for instance in during hiring and team structuring decisions.

Stakeholder representation competency requires the ability to gather, communicate, and balance the needs of other stakeholders, and accurately represent their views. This competency needs to be addressed by the Product Managers and Domain/ Application Requirements Engineers.

Analysis competency requires the ability to understand opportunities and their related stakeholder needs, and transform them into an agreed and consistent set of requirements. This competency needs to be addressed by Domain/Application Architects and Domain/ Application Requirements Engineers.

Development competency requires the ability to design and program effective software systems. This competency is needed to be addressed by Domain/Application Architects and Domain/Application Developers.

Testing competency requires the ability to test and verify a system that it meets the requirements. This competency is needed to be addressed by Domain/Application Test Engineers.

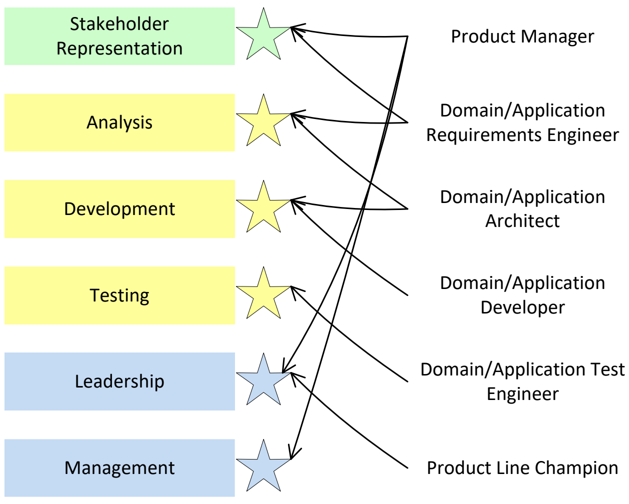

Stakeholder Representation
Analysis
Development
Testing
Leadership
Management
Product Manager
Domain/Application Requirements Engineer
Domain/Application Architect
Domain/Application Developer
Domain/Application Test Engineer
Product Line Champion

Figure 7. Mapping of SPLE roles to Essence Competencies

Leadership competency enables a person to inspire and motivate people to achieve a success and to meet their objectives. This competency is needed to be addressed by Product Managers and Product Line Champions.

Management competency requires the ability to coordinate, plan, and track the work done by a team. This competency is needed to be addressed by Product Managers.

4.4 Tracking SPLE Process

The Essence framework proposes to extend Essence Kernel to facilitate coordination, planning, and tracking in large software development projects [14]. Essence Kernel can be extended by defining sub-alphas. These sub-alphas can be used by different teams (e.g. domain and application teams) to plan activities and track their own progress. The checklists of these sub-alphas can also enable tracking interdependencies (for example, some domain requirements should be implemented before some application requirements). These sub-alphas drive the state changes on Kernel alphas. In this way, the overall progress can be tracked, bottlenecks can be discovered and proper concrete targets can be set by teams.

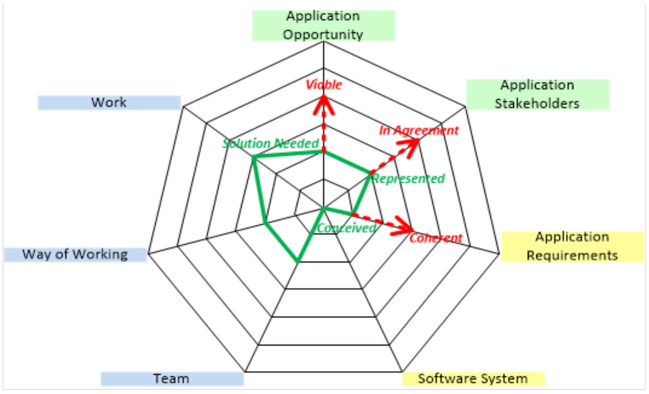


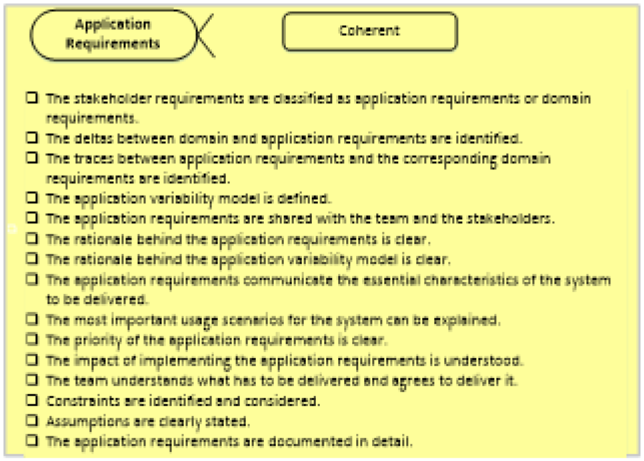
Figure 8. Radar chart for project progress

At any given point in time, the SPLE team can track the overall status of the SPLE effort. In Figure 8, we have depicted the status of one of the application's overall status on a radar chart. The radar chart shows the status of the application using the alphas. The application team sets a goal to advance in *Application Requirements*, *Application Opportunity* and *Application Stakeholder* states. To achieve this goal, *Develop the Application Requirements* activity must be applied.

Appendix A illustrates the *Develop the Application Requirements* activity on an *Activity Definition Card*. This card defines the initial states, goal states, involved work products, and associated roles related to the activity. *Develop the Application Requirements* activity may start when the *Application Stakeholders* is in *Represented* state, *Application Opportunity* is in *Solution Needed* state, and *Application Requirements* is in *Conceived* state. In addition to these, domain requirements should be at least in *Addressed* state. The card also shows that the *Application Requirements Engineer* who has the *Analysis* competency should conduct this activity. The other roles that are affected by this activity are the Customers, Product Manager, Domain Requirements Engineer and Application Architect. When the activity is completed, the *Application Stakeholders*, *Application Opportunity*, *Application Requirements* sub-alphas should reach the *In-Agreement*, *Viable*, *Coherent* states respectively.

Figure 9 shows the checklist for the *Coherent* state of the *Application Requirements* alpha, as an example, which can be consulted while the *Develop the Application Requirements* 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 [18] and modified to address SPLE concerns.

An *Activity Checklist* can be produced by aggregating the checklists for the *goal states* of the activity. For example, the checklist for the *Develop the Application Requirements* activity is given by the collection of the checklists for *Coherent* state of the *Application Requirements* alpha, the *In Agreement*, state of the *Application Stakeholders* alpha, the *Viable* state of the *Application Opportunity* alpha.


Application Requirements

Coherent

- ❏ The stakeholder requirements are classified as application requirements or domain requirements.
- ❏ The deltas between domain and application requirements are identified.
- ❏ The traces between application requirements and the corresponding domain requirements are identified.
- ❏ The application variability model is defined.
- ❏ The application requirements are shared with the team and the stakeholders.
- ❏ The rationale behind the application requirements is clear.
- ❏ The rationale behind the application variability model is clear.
- ❏ The application requirements communicate the essential characteristics of the system to be delivered.
- ❏ The most important usage scenarios for the system can be explained.
- ❏ The priority of the application requirements is clear.
- ❏ The impact of implementing the application requirements is understood.
- ❏ The team understands what has to be delivered and agrees to deliver it.
- ❏ Constraints are identified and considered.
- ❏ Assumptions are clearly stated.
- ❏ The application requirements are documented in detail.

Figure 9. Alpha state checklist

Adopting an SPLE process triggered an organizational change as well. We have addressed organization change issues via Team alpha of Essence Kernel. Hereby, Team performs, and plans work for producing software system by applying different ways of working. Further, sub-alphas and checklists can form a common ground for an organization regarding the organizational changes to be made. Moreover, the progress can be tracked, and potential problem areas can be detected.

5. DISCUSSION

RQ1: How to apply the Essence framework to model an SPLE process?

In this paper, we have provided our experiences in mapping an SPLE process to the Essence framework and the required enhancements. So far, the Essence framework has been applied to single system development methods and not yet for SPLE methods. To the best of our knowledge, this is the first study to map an SPLE method to the Essence framework. Modeling SPLE methods appears to be an important test to evaluate the expressiveness of the Essence framework. This is due to the complexity and the two-life cycle nature of the SPLE methods.

An important aspect of the SPLE methods is the distinction between application engineering and domain engineering, and herewith the two-life cycle approach. The Essence framework has been primarily applied to single system development methods and did not consider two life cycle methods. One of the fundamental decisions was either mapping the domain engineering and application engineering instances as separate Essence instances or mapping the whole SPLE effort as a single Essence instance. To capture the interactions between application engineering and domain engineering, we decided to map both life-cycles to a single essence instance. If we would map the two-life cycle process to separate Essence instances then we would miss the details regarding the communication between the application engineering and domain engineering processes. Modeling this interaction is important to explicitly describe the reuse relations. Hence, the decision to have one instance instead of two instances is based on the reason to model the inherent properties of the two-life cycle process.

As a SPLE methodology, we have chosen Pohl et al. [22] for mapping to the Essence framework. There are indeed several different SPLE methods and selecting an SPLE method for our study was an initial question. The reason for selecting Pohl et al.[22] over other SPLE approaches is that; Pohl et al. [22] provides a very detailed explanation of the SPLE process, which made it feasible to map to the Essence framework. Here our aim was not to compare all SPLE approaches and select the best one, but rather explore the benefits of applying Essence framework on SPLE. In this study, we have provided an approach for mapping SPLE methods to the Essence framework. The approach is generic and can be applied for modeling other SPLE methods. In theory, we can combine different SPLE methods to enhance the systematic reuse goals. The Essence framework can be used as a vehicle to perform comparative analysis of different SPLE methods as well. This is considered as part of our future work.

RQ2: What are the lessons learned with respect to modeling a two-life cycle process with the Essence framework?

Within the industrial context of Havelsan, it was important to know whether the Essence framework is expressive enough to model the SPLE process that is required. From our experiences as reported in this paper, we can state that the Essence framework appeared indeed to be largely expressive to model the selected SPLE method. In general, we could map the SPLE method elements to the Essence framework elements. For some of the elements, though, we had to extend the Essence framework elements. Since adding new method elements is supported by the Essence framework, we did not face with too many difficulties.

Besides of the end-result, we observed that the process for modeling the method per se provides a lot of insight in the SPLE method that was modeled. We have conducted several tutorial sessions on SPLE. Although these sessions significantly increase the knowledge on SPLE for both process designers and SPLE project team, the mapping process enforced to fully understand the method elements to map these to the Essence framework elements. This appeared to be an important exercise for the process designers at Havelsan. "The whole mapping exercise helped us to understand the SPLE process. We could not derive this knowledge directly from the textbooks" was one of the comments of the managers. During the activity, novel ideas were triggered on how to solve areas that are not addressed by SPLE, and solutions were searched to complement the method with elements of other software methods. This on its turn supported the idea of method tailoring that focuses on enhancing and adapting existing methods for adapting to specific contexts and criteria.

A model is not only good for enhancing understandability and assessment of the method; it can also help to guide the development process. In this case, the model of the SPLE method with the Essence framework helped to trace the method steps during the actual operation of the method. The defined cards based on the modeled SPLE method helped to track the progress of the project. Using Essence implies frequently moving the cards around a table to facilitate discussion between team members. From our experiences, the use of physical cards supports the engagement of team members in the SPLE project, however this improvement was very limited. Even though a tool for describing a method modeled using Essence Framework is in place, to the best of our knowledge, no tool support is present for applying a method modeled using Essence Framework in a specific project. According to the feedback from the SPLE project team members, we started to develop a monitoring tool support that will integrate Essence states with our Application Lifecycle Management toolset within the company.

Our study had the benefit for supporting the understanding of SPLE method but also led to the experience with the Essence framework. This paved the way for exploring the mapping of other methods applied within Havelsan. In this context, a manager commented "This exercise was very helpful, but we also adopt other software development methods in the company ranging from plan-driven methods to agile methods. Actually, we need a common framework to track all the different method steps in the different projects. Could we use the Essence framework for this purpose?" We believe that this is indeed possible but consider this as a future work.

RQ3. What are the identified limitations of the SPLE process after modeling it with the Essence Framework?

We have performed unstructured interviews with the stakeholders (project manager, technical lead, 2 process designers, and 2 software engineers) and ask their opinions about the essentialization process. Based on the stakeholders' comments and interviews, the mapping process provided an increased understanding of the SPLE methods. On the other hand, it also helped to look critically at the SPLE method and support the reviewing of the method for practical use. As stated before, during the mapping process we can be confronted with the lack of some of the Essence framework elements in the SPLE method. This trigger the discussion on whether the SPLE method needs to be enhanced to address a missing Essence framework element. Neither the Essence framework nor our systematic mapping approach coerces this extension of the method, but at least it paves the way for a sound critical review and as such supports further enhancements. In this context, in the literature we can identify several efforts for integrating, for example, agile software development methods with SPLE [17] [7]. The Essence framework could be adopted to support these endeavors in a systematic manner.

6. RELATED WORK

Essence specification published by Object Management Group (OMG) [18] includes some demonstrations for mapping practices to the Essence framework, such as Scrum, user story, Unified Process, and Waterfall lifecycle. Park provided a complete mapping of Scrum to the Essence framework [20] using Scrum Guide [24] as a method specification. In this study, Park provided a method composition case using Scrum, XP, and DevOps. In our earlier work, we provided an approach for mapping development methods to the Essence framework and a partial mapping of Nexus to the Essence framework [10]. In [9], we applied this approach to build an initial practice library for Internet of Things system development methods using Essence framework.

The Software and Systems Process Engineering Metamodel (SPEM) is a meta-model of process engineering, that is used to define development processes and their components [19]. Both SPEM and Essence framework provide a language to define methods. In [8], the authors present a comparison of Software & Systems Process Engineering Metamodel (SPEM) 2.0 and the draft version of Essence Specification 1.0. The Essence framework, however, also provides a generic domain model of software engineering defined in Essence language (e.g. Essence Kernel). This common domain model forms a base to understand, compare, and combine practices and methods. Moreover, Essence framework emphasizes the importance of tracking progress and health of a project using alphas and sub-alphas.

Related to the tracking the health of a software project, SEI framework [6], defines Monitor pattern. Monitor pattern has the responsibility of measuring and maintaining the course and operation of an established, running product line. The Monitor pattern includes practice areas that serve to monitor an ongoing product line effort and apply course corrections to keep activities on track. There are two groups of practice areas that address the solution and provide the structure for the Monitor pattern. The listen group includes the Customer Interface Management, Measurement and Tracking, Organizational Risk Management, and Technical Risk Management, whereas the response group includes Organizational Planning, Technical Planning, and Process definition practice areas which are defined in the SEI framework [1]. Although this gives an initial guidance on how to operate and track the status of a product line, they are not as descriptive as the Essence Framework's guidance on tracking alphas.

7. CONCLUSION

In this study, we proposed an approach that can be used to model SPLE methods using the Essence framework. The Essence framework seemed to be expressive to model the selected SPLE method, but we have also identified several challenges due to the two-life cycle approach of SPLE. In addition, we have observed, several aspects that are important for SPLE, such as organizational concerns, are not directly addressed in the Essence framework. Nevertheless, we can state that the exercise of mapping the selected SPLE method to the Essence framework enhanced the understanding and provided critical insight in both the definition and the operation of the method. In addition, the overall process helped to explicitly identify the points for improvement of the SPLE method and paves the way for tailoring SPLE methods. In our future work, we will build on our current experiences and elaborate on modeling and tailoring SPLE methods.

REFERENCES

[1] A Framework for Software Product Line Practice, Version 5.0: 2007. *http://www.sei.cmu.edu/productlines/framework.html*.

[2] America, P. et al. 2000. CoPAM: A Component-Oriented Platform Architecting Method Family for Product Family Engineering. *Software Product Lines*. Springer US. 167–180.

[3] Bayer, J. et al. 1999. PuLSE: a methodology to develop software product lines. *SSR '99 Proceedings of the 1999 symposium on Software reusability* (1999), 122–131.

[4] Brinkkemper, S. 1996. Method engineering: engineering of information systems development methods and tools. *Information and Software Technology*. 38, 4 (Jan. 1996), 275–280. DOI:https://doi.org/10.1016/0950-5849(95)01059-9.

[5] Clarke, P. and O'Connor, R. V. 2012. The situational factors that affect the software development process: Towards a comprehensive reference framework. *Information and Software Technology*. 54, 5 (May 2012), 433–447. DOI:https://doi.org/10.1016/j.infsof.2011.12.003.

[6] Clements, P. and Northrop, L. 2001. *Software Product Lines: Practices and Patterns*. Addison-Wesley.

[7] Díaz, J. et al. 2011. Agile product line engineering - A systematic literature review. *Software - Practice and Experience*. 41, 8 (2011), 921–941. DOI:https://doi.org/10.1002/spe.1087.

[8] Elvesæter, B. et al. 2013. A Comparison of the Essence 1.0 and SPEM 2.0 Specifications for Software Engineering Methods. *Proceedings of the Third Workshop on Process-Based Approaches for Model-Driven Engineering* (New York, New York, USA, 2013).

[9] Giray, G. et al. 2017. Adopting the Essence Framework to Derive a Practice Library for the Development of IoT Systems. *Connected Environments for the Internet of Things: Challenges and Solutions*. Z. Mahmood, ed. Springer International Publishing. 151–168.

[10] Giray, G. et al. 2016. Systematic Approach for Mapping Software Development Methods to the Essence Framework. *The 5th International Workshop on Theory-Oriented Software Engineering* (2016), 26–32.

[11] Gomaa, H. 2005. Designing Software Product Lines with UML. *Engineering*. April (2005), 160–216. DOI:https://doi.org/10.1109/SEW.2005.5.

[12] Havelsan Corporate Web site: *www.havelsan.com.tr/SirketProfili/ENDefault.aspx*.

[13] Henderson-Sellers, B. and Gonzalez-Perez, C. 2011. Towards the Use of Granularity Theory for Determining the Size of Atomic Method Fragments for Use in Situational Method Engineering. *IFIP Advances in Information and Communication Technology*. 49–63.

[14] Jacobson, I. et al. 2013. *The Essence of Software Engineering*.

[15] Kalus, G. and Kuhrmann, M. 2013. Criteria for software process tailoring: a systematic review. *Proceedings of the 2013 International Conference on Software and System Process - ICSSP 2013* (2013).

[16] Larman, C. 2004. *Applying UML and Patterns: An Introduction to*

Object-Oriented Analysis and Design and Iterative Development (3rd Edition). Prentice Hall, Inc.

[17] Mohan, K. et al. 2010. Integrating Software Product Line Engineering and Agile Development. *IEEE Software*. 27, 3 (2010), 48–55.

[18] Object Management Group 2015. Essence - Kernel and Language for Software Engineering Methods. *OMG*. Version 1.1 (2015). DOI:https://doi.org/http://www.omg.org/spec/Essence/1.0/PDF/.

[19] Object Management Group 2008. Software & Systems Process Engineering Meta-Model Specification. Version 2 (2008).

[20] Park, J.S. et al. 2016. Scrum Powered by Essence. *ACM SIGSOFT Software Engineering Notes*. 41, 1 (2016), 1–8. DOI:https://doi.org/10.1145/2853073.2853088.

[21] Péraire, C. 2013. A Step Forward in Software Engineering Education : Introducing the SEMAT Essence Framework. *Latin American Congress on Requirements Engineering and Software Testing (LACREST)* (Medelin, 2013).

[22] Pohl, K. et al. 2005. *Software product line engineering: foundations, principles, and techniques*. Springer.

[23] Schmid, K. and Verlage, M. 2002. The Economic Impact of Product Line Adoption and Evolution. *IEEE Software*. 19, 4 (2002), 50–57.

[24] Schwaber, K. and Sutherland, J. 2011. The scrum guide. *Scrum. org, October*. 2, July (2011), 17. DOI:https://doi.org/10.1053/j.jrn.2009.08.012.

[25] Software Engineering Method and Theory: *http://semat.org/*. Accessed: 2016-03-26.

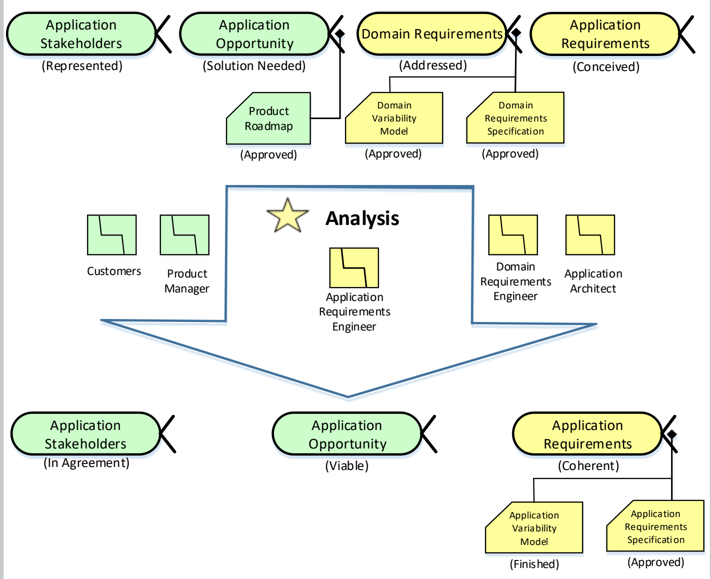
[26] Tüzün, E. et al. 2015. Empirical evaluation of a decision support model for adopting software product line engineering. *Information and Software Technology*. 60, (2015). DOI:https://doi.org/10.1016/j.infsof.2014.12.007.

[27] Weiss, D.M. and Lai, C.T.R. 1999. *Software Product-Line Engineering: A Family-Based Software Development Process*. Addison-Wesley Professional.

Appendix A. Activity Definition Card for "Develop the Application Requirements" Activity

Understand the Requirements

Develop the Application Requirements



Develop the Application Requirements to provide a consistent description of the essential characteristics of the application.

This activity is started when;

- Application Stakeholders : Represented
- Application Opportunity : Solution Neded
- Domain Requirements : Addressed
- Product Roadmap : Approved
- Domain Variability Model : Approved
- Domain Requirements Specification: Approved

Roles :

- Main Role : Application Requirements Engineer
- Related Roles : Customer, Product Manager, Domain Requirements Engineer, Application Architect

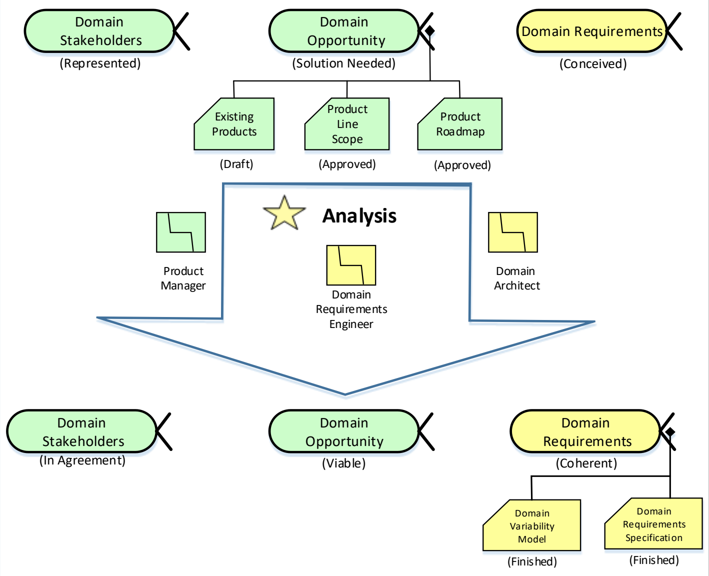
This activity is completed when;

- Application Requirements : Coherent
- Application Stakeholders : In Agreement
- Application Opportunity : Viable
- Application Variability Model : Finished
- Application Requirements Specification: Approved

Appendix B. Activity Definition Card for "Develop the Domain Requirements" Activity

Understand the Requirements

Develop the Domain Requirements



Develop the Domain Requirements to provide a consistent description of the essential characteristics of the platform.

This activity is started when;

- Domain Stakeholders: Represented
- Domain Opportunity: Solution Neded
- Domain Requirements: Conceived
- Product Line Scope: Approved
- Product Roadmap:Approved

Roles :

- Main Role : Domain Requirements Engineer
- Related Roles : Product Manager, Domain Architect

This activity is completed when;

- Domain Requirements : Coherent
- Domain Stakeholders : In Agreement
- Domain Opportunity : Viable
- Domain Requirements Specification: Finished
- Domain Variability Model : Finished