Software Startup ESSENCE – How Should Software Startups Work?

Kai-Kristian Kemell[1[0000-0002-0225-4560]], Anh Nguyen-Duc[2[0000-0002-7063-9200]], Xiaofeng Wang[3[0000-0001-8424-419X]], Juhani Risku[1[0000-0002-0587-4431]], and Pekka Abrahamsson[1[0000-0002-4360-2226]]

[1] University of Jyväskylä, 40014 Jyväskylä, Finland
{kai-kristian.o.kemell|pekka.abrahamsson|juhani.risku}@jyu.fi
[2] University of Southeast Norway
angu@usn.no
[3] Free University of Bozen-Bolzano, 39100 Bozen-Bolzano, Italy
xiaofeng.wang@unibz.it

Executive Summary: Software startups need to work in a systematic fashion just like mature organizations. However, existing software engineering methods and practices are not aimed at software startups. They do not account for the business aspect of startups and may not be well-suited for software startups in general. The Lean Startup Methodology on the other hand contains some useful practices for software startups but is nonetheless impractical, offering little in the way of telling you what to do.

Software startups are thus required to tailor their own method. Currently, many software startups simply work ad hoc or use various agile methods and practices. In terms of Agile methods and practices, little consensus exists between startups. In this chapter, we discuss methods and method tailoring. We give guidelines on how to create your own way of working and recommend a tangible tool for doing so: the Essence Theory of Software Engineering.

1 Introduction

Software startups should work in a systematic fashion in order to be effective, according to Eric Ries [10], a high-profile expert on startups and a startupper himself. In practice, working systematically means utilizing a method that is known and understood by the team. A method tells you how to work.

The issue here is that little exists in the way of methods for software startups. There are no widely known software engineering methods specifically aimed at software startups. While software startups can utilize the numerous existing software engineering practices and methods as they wish, they may not be well-suited for software startups. Similarly, the Lean Startup Methodology [10] offers startups some ideas, principles and practices to utilize, but does not directly tell you how to work and what to do. In other words, it is not practical. Each startup is thus responsible for creating its own method, or way of working.

In this chapter, we will discuss how a way of working can be devised by a software startup and discuss The Essence Theory of Software Engineering as a tool for doing so. The goal of this chapter is to underline the importance of actively reflecting on your way of working and to provide some guidelines for doing so.

Though software startups have been extensively studied in the academia, little research currently exists on this point of view [12]. This chapter presents some preliminary results from an on-going study in the form of a deck of practice cards presented in the second-to-last section. In addition, we provide some tools to utilize in this context based on past studies as well.

2 How Should Software Startups Work? Ideas and the Current State of Practice

A method is your way of working. It describes the way you work, should work, or want to work according to the description. A method can be broken down into parts that can be referred to as practices. Practices are micro-level methods that, again, describe some parts of the work process. Together, practices form methods. In terms of software engineering, a method is e.g. Scrum[1][8] while a practice is e.g. pair programming. A (communication) practice can also be something as simple as using a specific software to communicate in a specific way inside the team.

Software startups develop software and can thus utilize various existing models, methods and practices from the field of software engineering (SE). SE methods are numerous, ranging from the early so-called waterfall models to the currently fashionable, highly diverse Agile methods. Agile itself is not a method but a set of principles outlined in the Agile manifesto[2], and the number of methods and practices that can be considered Agile is vast [1].

Existing SE methods are not specifically aimed at startups. In fact, on the contrary, they are generally intended for project use inside mature organizations. These SE methods thus focus entirely on SE, omitting the business aspect that is vital for startups. In these more mature organizations, the software developers are not interested in the work of the financial department and often have to care little for how the software is marketed towards its intended customers as that is also the job of another department. In a startup, on the other hand, the far smaller organization size often makes these roles more intertwined especially early on in the life of the software startup. Therefore, while these methods can still be utilized by startups, they can only help startups with their software engineering, which is only a portion of the work of a startupper.

In practice, software startups utilize a wide variety of methods and practices for developing software, based on a study by Paternoster et al. [9]. According to the study, software startups largely utilized various agile methods and practices or simply worked ad hoc. Little consensus on what the best method or practice(s) for software engineering would be was found between the software startups studied by Paternoster et al. [9]. This

[1] https://www.scrum.org/
[2] https://agilemanifesto.org/

further highlights the idea that startups, much like mature organizations, should concern themselves with finding or creating a method that suits their work in particular.

Aside from SE methods, there is little currently available in the form of more comprehensive methods for software startups. The lean startup, for example, is not a method that tells you how to work. It is more akin to the Agile manifesto in that it merely presents some principles associated with the idea of a lean startup. Some practices such as the Build-Measure-Learn loop are considered important or even vital for startups wishing to be lean, although they still offer little in terms of directly telling what to do.

Aside from, and including, the Build-Measure-Learn loop, various good practices recommended by different experts and startuppers alike do exist, even if they do not come bundled into any method(s). These startup practices can help you carry out some parts of your work better. Popular practices associated with startups include the aforementioned Build-Measure-Learn loop, the Five Whys, and the Minimum Viable Product. For example, the Five Whys practice is about understanding the root of a problem.

An example of this practice is presented as follows (taken from Wikipedia[3]):

Problem: The car won't start
Why? The battery is dead.
Why? The alternator is not functioning
Why? The alternator belt was broken
Why? The alternator belt was old and had not been replaced
Why? The vehicle had not been maintained properly

While such practices are useful, they do not offer comprehensive tips or tricks for success. They do not offer roadmaps or tell you what to do next. They can direct your way of working into the right direction but it is up to you to make the most of them. They can be valuable when included into your method, but it is your job to create that method. Startups, just like mature organizations, should concern themselves with working in a systematic fashion in order to be effective [10].

3 How to Develop Your Method or Way of Working?

Whether you are working ad hoc, deciding on how to do certain things as new tasks arise, or you are utilizing a formal method like Scrum, you have a way of working. A way of working comprises all of your work practices from the way your team communicates to what software tools to you use to perform various tasks. Together, these various practices for a method, or a way of working.

As we established in the previous section, there is currently no universal recipe for success for software startups. Software startups can utilize existing software engineering methods and practices but they do not tackle the business aspect of a startup. Business-related methods for startups are scarce, and while various good practices recommended by practitioners exist (the Five Whys etc.), they do not tell you how you should

[3] https://en.wikipedia.org/wiki/5_Whys - as it was on the 13th of March 2019

be working outside that one practice. These existing SE methods and various practices can, however, be utilized and combined in order to create your own method.

Your way of working encompasses everything you do in your startup, from your communication practices and your work hours to the tool(s) you use to develop software. How do you communicate: where and when do you have meetings, how do you communicate online when not in the office? Which tools do you use while developing your software?

Creating a way of working is not a one-time effort, as your way of working is not static. As new tasks arise and old ones are completed, what you work on is constantly changing. Thus, in order to work effectively, you should actively reflect on your way of working as you progress. Practices that were useful at one point may become less relevant and ultimately it might be better to phase them out completely. This is something that generally happens naturally, but it can be even more effective to do it consciously and to discuss it within the team.

Indeed, improving your way of working is highly related to consciously seeking to learn from what you are doing. Pay mind to what you are doing and why you are doing those things. Use metrics to measure how some of your practices work and then compare them to alternatives (we talk more about using metrics in the chapter titled "100+ Metrics for Software Startups – Common Practices of Using Metrics").

> **Practical Example:** You are starting a display advertising campaign. Rather than putting all of your eggs into one basket, you may wish to try different channels (e.g. Google, Facebook etc.) simultaneously. Track where the traffic is coming from. How much did you spend on each channel and how much did each channel bring in traffic in comparison? Pay mind to your advertising settings as well: whom are you targeting, which countries are you targeting, when are your ads showing for the people etc.
>
> Then, based on the data, pick the best channel(s) and focus on them. Try out two different ads in the same channel over the course of e.g. two weeks, one week for each. Which one brought in the most traffic? In this fashion, you can utilize metrics to decide which practice works best for your startup.

If you decide to utilize existing practices and methods, it is also worthwhile to evaluate them as you use them. You may find that modifying them rather than following them by the book works better for you.

4 Using the Essence Theory of Software Engineering to Describe and Improve Your Way of Working

The Essence Theory of Software Engineering [4] is a modular framework for constructing your own way of working. In short, Essence gives you a foundation that you can use as a foundation to construct your own way of working. This foundation comes in the form of a so-called kernel. The elements in this kernel can then be expanded upon

using the Essence language in order to tailor your own way of working. Essence supports the use of any existing software engineering method or practice or combination of such [8].

4.1 Why is This Relevant?

Essence provides you with a tool that can help you systematize your way of working [5]. Communication tends to be the most important problem in most projects and Essence is a tool meant to facilitate and improve communication. You may feel that modeling your way of working using Essence is stating the obvious that your team already knows. This is not the case at all, as the point of Essence is not to model our way of working for the sake doing so for busywork.

By modelling your way of working using Essence you present it in a formal, visual manner. This can be thought provoking and offers a good basis for discussion within the team. Does everyone agree that this is a good way to be doing things? If not, why? And how should it be changed?

Aside from using Essence to build and describe your way of working, Essence is useful for progress management [5]. The alphas and alpha states can help you better understand where you currently stand in terms of the software you are developing or your startup in general. Again, though it may initially feel like extra work to establish something you already know, your team may in fact not fully be on the same page about your progress.

Essence is not something one person in your team should utilize alone in order to manage your progress or to model your way of working. It is a communication tool and it should be used like one. Essence facilitates communication within the team and is helpful in communicating your progress and/or your way of working in a clear manner. It can also serve as a way to motivate your team members to think about they own work practices as individuals through your team meetings.

4.2 Quick Overview of Essence

The kernel is the foundation that contains the building blocks you can use to build your own method. It contains the basic elements present in every single software engineering project. The kernel consists of three different types of elements split into three categories called *Areas of Concern*. These areas of concern are denoted by three colours:

- *Customer* (Green)
- *Endeavor* (Yellow)
- *Team* (Blue)

These categories each contain three types of elements:

- *Alphas*, a.k.a. Things to Work With
- *Activity spaces*, a.k.a. Things To Do
- *Competencies*, i.e. the skills required to carry out tasks

The core of the kernel (seen below in Fig. 1) is formed by the *alphas*, of which there are seven. These seven alphas are present in every software engineering project and thus they are the essentials. In other words: most projects have other things to take into

account as well, but every project includes at least these elements among other things. The alphas are colour-coded according to the above split into areas of concern, also denoted in the kernel itself.

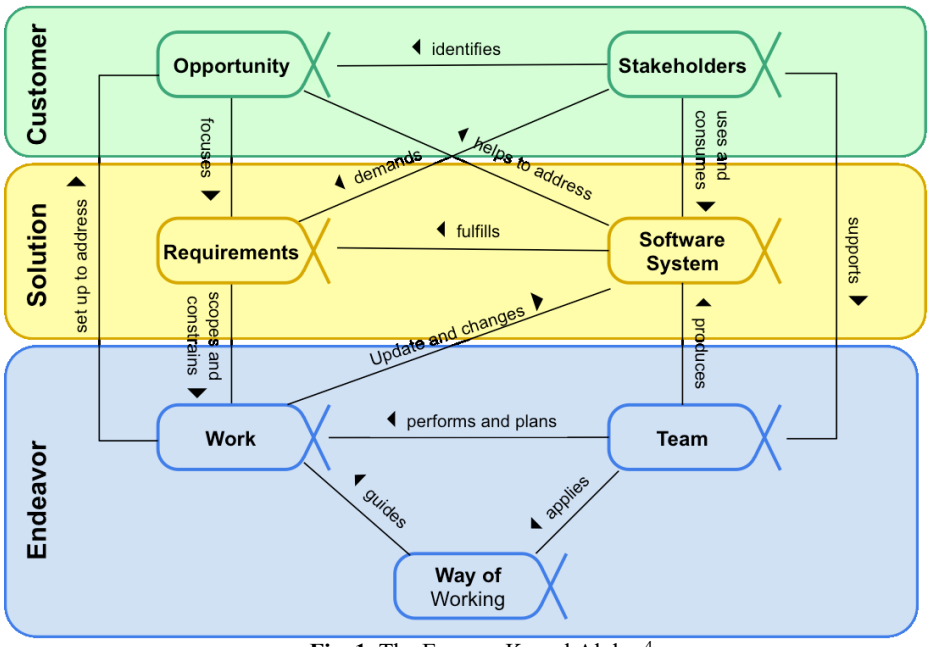

Fig. 1. The Essence Kernel Alphas[4]

Each of the seven alphas has *alpha states* which are meant to help track progress in your startup. For example, the *requirements* alpha, below, starts with the requirements being outlined and ends when the software has fulfilled them. Every software has requirements even if they do not directly come from a customer commissioning it.

In addition to the alphas, the kernel contains *activity spaces*, which contain some of the things that should be done in every software engineering project. As was the case with alphas, every project contains other things to do as well, but these are the essential things that should be done in every project. Finally, *competencies* are the skills required to carry out the tasks. For example, if you are developing a game using Unity, a developer joining your startup should probably be familiar with Unity, or at least be ready to learn it. The kernel contains some competencies that are required in every project.

The kernel contains only the essentials present in every project, but every project has its own elements aside from the essentials. For the kernel to be more useful for you, you can utilize the Essence language to add the unique characteristics of your software startup to it. Add your own alphas, describe your own practices, and draw your own method using the graphical elements of Essence.

[4] http://semat.org/quick-reference-guide

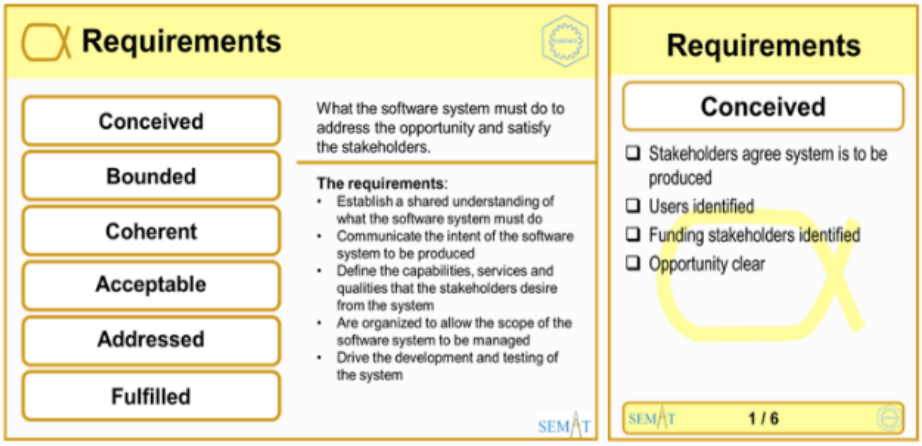

Fig. 2. The Requirements Alpha and Its First State[5]

4.3 Essentializing Your Way of Working

The act of describing your work practices using Essence is called essentializing them. This is done by using the Essence language to describe the practices. Essentializing a practice has three steps:

1. Identifying the elements. Use the element types of the Essence language to describe the elements present in the practice. The output is a diagram. (See Fig. 3. below)
2. Drafting the relationships between the elements. Make practice cards.
3. Providing further details. Add descriptions, references etc. to the cards.

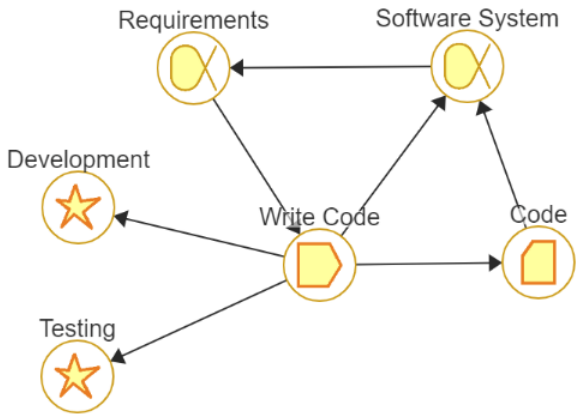

Fig. 3. Pair Programming Described with Essence, Drawn with Essencery[6]

[5] http://semat.org/quick-reference-guide
[6] Essencery.com

In addition to the alphas, activity spaces, and competencies discussed in the previous sub-chapter, the Essence language contains a few other elements. These are *activities*, *work products*, and *patterns*. Activities, like activity spaces, are things to do, but activity spaces are placeholders and can act as containers or umbrellas for groups of activities. Work products are tangible things you create as you work, e.g. meeting notes or PowerPoint presentations about your software idea. Finally, patterns are arrangements of other elements presented in the language, such as roles (programmer, designer etc.).

How much you conform to the Essence language as you essentialize your practices and describe your way of working is up to you. Ultimately, it is not necessarily important that you are fully conformant to the language. Essence is a communication tool and if what you are doing facilitates communication in your team while successfully describing what you wish to describe with the language, you have reached your goal.

You do not have to precisely tie the practices and practice cards to e.g. the alpha states, especially at first. Once you have devised a method that really works for your team, one that you can keep using in the future, you may consider doing so. Being fully conformant with Essence takes time and if you are continuously changing your way of working, it may be better to settle for a lower conformance level in your practice descriptions. You do not have to utilize Essence in its entirety, by the book, for it to help you. In the next sub-section, we will provide you with links and resources that can help you get started with Essence.

4.4 Getting Started with Essence

If you are interested in Essence, there are various resources available online that can give you a more in-depth overview of it, as well as tools that can help you get started with it more easily. Below is a list of resources that you can utilize to get started with Essence:

- **OMG Standard for Essence.** The full documentation for Essence.[7] [7]
- **Quick Reference Guide.** A formal guide for getting started with Essence.[8]
- **SematAcc.** A digital tool for tracking alpha states.[9][3]
- **Essencery.** A digital tool for drawing graphs using the Essence language.[10][2]
- **Ivar Jacobson Practice Card Library.** A database of various existing software engineering practices made into Essence practice cards. You can get ideas for new practices from here and it can save you lots of time when making practice cards.[11]
- **Essence of Software Engineering – The Board Game.** A board game to teach you the idea of Essence.[12][6]

[7] https://www.omg.org/spec/Essence/
[8] http://semat.org/quick-reference-guide
[9] https://ineed.coffee/projects/sematacc-2/ – can be downloaded
[10] Essencery.com – used directly in the browser
[11] https://practicelibrary.ivarjacobson.com/start – requires registration
[12] https://doi.org/10.6084/m9.figshare.8052653.v1 – permanent link to all the game materials

4.5 Essentials for Early-Stage Software Startups: An Example of Using Essence

Using Essence as a base, we have developed a set of practices intended to help early-stage software startups (Fig. 4). Though later on startups branch out and become very different, very early on in their lives they share more similarities. Building on these similarities, we have devised a set of Practice Cards for use with Essence.

Fig. 4. The Card Deck

These cards are intended to guide an early-stage software startup from idea generation to the early stages of developing their first product. Each card describes a practice

that is intended to help early-stage startups progress. As an early-stage startup, the deck can:

- Offer you an additional starting point for essentializing your way of working
- Give you ideas on what to do next
- Give you ideas for new practices

This set of cards was developed for a university course and as a part of an on-going study. The deck is being improved iteratively and this is the result of the first iteration, used as a teaching tool in a course teaching startup entrepreneurship at the University of Jyväskylä. These cards have not yet been formally bound to the Essence kernel and act as a stand-alone tool as well. There are 12 cards in total in this deck.

As these cards were developed for a university course on Software Startups, some of the cards contain lingo indicating this original use purpose. These cards are available in their entirety on FigShare (https://doi.org/10.6084/m9.figshare.8378900.v1).

5 Methodology: Creating the Card Deck

The card deck presented in the preceding section (4.5) was created as a part of an on-going study on Essence in the context of software startups. The idea of the study is to ultimately tailor a version of the Essence kernel for software startups. In the process, we aim to create a practice card deck for early-stage software startups. The first iteration of said practice card deck was featured in this chapter.

The first iteration of the card deck was based on existing academic and practitioner literature on software startups. For example, as a basis for the deck, we used the life-cycle stages discussed by Wang et al. [13] in their study. According to this view, a software startup begins with problem definition as they come up with a problem to solve, or a need to address. They then move on to problem validation as they try to ascertain that the problem or need is in fact a real one someone wants to have addressed and is perhaps willing to pay to have addressed. Having validated the problem, the next stage is to then come up with a solution to carry out the idea: solution definition. Once the solution has been defined, it, too, must be validated. The solution is validated in practice when (if) the startup becomes a functioning firm, at which point the software startup slowly grows into a mature organization and ceases to be a startup.

Building on this idea of four life-cycle stages, we devised a card deck depicting key practices associated with this process. These practices are well-established ones discussed in academic literature and by experts alike, such as pivoting or validation using an MVP. These cards are still being developed and thus more practices are likely to be added in future versions.

This iteration of the deck was deployed in a university course on software startup entrepreneurship at the University of Jyväskylä. In the course, teams of 3 to 5 students were to found a software startup based on a new or existing idea. We gave each team a physical deck of the cards to use. The cards were intended to guide them by highlighting various actions they should perform during the course, and more importantly, to progress as startups. Data was collected on their experiences with the cards by having the

teams report their use of the cards by writing on them. This data will be used to improve the cards for future iterations.

Finally, it should be noted that the cards were devised with the educational setting in mind. Thus, some of the cards contain some material that may not be fully relevant to a software startup out on the field. For example, the "Startup Spirit" card discusses treating the startup as a real startup and not simply a course project, which is something that does not directly concern most startups.

6 Conclusions: Managerial Implications

In this chapter, we have discussed the importance of working in a systematic fashion and constantly improving your way of working. Though software startups should also seek to work in a systematic fashion, methods and practices aimed at software startups are rare in software engineering. Existing methods are devised with large, established organizations in mind. They are aimed at project use and focus exclusively on the software engineering aspect of the project, leaving out the business aspect that is just as important for a software startup.

Similarly, little exists in the way of startup-oriented business practices or methods. Though some practices recommended for startups such as the Five Whys and the Build-Measure-Learn loop discussed in the context of the Lean Startup methodology exist, even the lean startup methodology does not offer startups actionable and practical advice on how to work. Thus, it is up to each software startup to devise their own method.

In this chapter, we discussed how a method or way of working should be devised, and recommended Essence as one tool for doing so. Though no comprehensive methodologies for software startups exist, startups should nonetheless aim to work in a systematic fashion [10]. In devising their own method, software startups can utilize these various existing practices in creating a method by using them as building blocks. Existing methods such as Scrum can be tailored in this fashion to better suit the context and needs of a startup as opposed to a large, established organization.

Once a method is created, it should be iteratively developed. This can be done by e.g. measuring the effectiveness of individual work practices, as we discussed in section 3 of this chapter. The method, and the practices it consists of, should be discussed within the team in order to make sure everyone is on the same page. This way, everyone can provide their point of view on the way the team works – or should work.

Key Take-Aways

- Startups, just like more mature organizations, should work in systematically
- Model your way of working in order to critically evaluate it
- Continuously pay mind to how you work and look for points of improvement
- Discuss your way of working with your team in order to improve it
- Use metrics to objectively evaluate your work practices (check out the chapter "100+ Metrics for Software Startups – Common Practices of Using Metrics)
- You can use Essence to model your way of working

- If you are interested in Essence, use the tools suggested in 4.4 to get started
- The deck of cards we presented can give an early-stage software startup new ideas for work practices and can be used as a base to build a way of working on using Essence

References

1. Abrahamsson, P., Salo, O., Ronkainen, J., and Warsta, J.: Agile Software Development Methods: Review and Analysis. Otamedia: VTT Publications, 478 (2002).
2. Evensen, A., Kemell, K., Wang, X., Risku, J., and Abrahamsson, P.: Essencery - A Tool for Essentializing Software Engineering Practices. arXiv preprint https://arxiv.org/abs/1808.02723?context=cs (2018).
3. Graziotin, D., and Abrahamsson, P.: A Web-based modeling tool for the SEMAT Essence theory of Software Engineering. Journal of Open Research Software, 1 (2013).
4. Jacobson, I., Ng, P., McMahon, P. E., Spence, I., and Lidman, S.: The Essence of Software Engineering: The SEMAT Kernel. ACMQueue, 10, pp. 40-52 (2012).
5. Kemell, K. K., Nguyen-Duc, A., Wang, X., Risku, J., and Abrahamsson, P.: The Essence Theory of Software Engineering - Large-Scale Classroom Experiences from 450+ Software Engineering BSc Students. In Proceedings of the 2018 International Conference on Product-Focused Software Process Improvement (PROFES2018) (2018).
6. Kemell, K. K., Risku, J., Evensen, A., Dahl, A. M., Grytten, L., Jedryszek, A., Rostrup, P., Nguyen-Duc, A., and Abrahamsson, P.: Gamifying the Escape from the Engineering Method Prison - An Innovative Board Game to Teach the Essence Theory to Future Project Managers and Software Engineers. In Proceedings of the 2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC), Stuttgart (2018).
7. Object Management Group: Essence – Kernel and Language for Software Engineering Methods. Version 1.1. http://semat.org/essence-1.1, last accessed 2018/05/28.
8. Park, J. S., McMahon, P. E., and Myburgh, B.: Scrum Powered by Essence. ACM SIGSOFT Software Engineering Notes, 41(1), pp. 1-8 (2016).
9. Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T. and Abrahamsson, P.: Software development in startup companies: A systematic mapping study. Information and Software Technology, 56, pp. 1200-1218 (2014).
10. Ries, E.: The Lean Startups: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. New York: Crown Books (2011).
11. SEMAT: SEMAT and Essence – What is it and why should you care? http://semat.org/what-is-it-and-why-should-you-care-, last accessed 2018/05/20.
12. Unterkalmsteiner, M. Abrahamsson, P. Wang, X. F. Nguyen-Duc, A. Shah, S. Bajwa, S. S. Baltes, G. H. Conboy, K. Cullina, E. Dennehy, D. Edison, H. Fernandez-Sanchez, C Garbajosa, J. Gorschek, T. Klotins, E. Hokkanen, L. Kon, F. Lunesu, I. Marchesi, M. Morgan, L. Oivo, M. Selig, C. Seppänen, P. Sweetman, R. Tyrväinen, P. Ungerer, C. Yagüe, A.: Software Startups - A Research Agenda. e-Informatica Software Engineering Journal, 10(1), pp. 89-123 (2016).
13. Wang, X., Edison, H., Bajwa, S. S., Giardino, C., and Abrahamsson P. (2016). Key Challenges in Software Startups Across Life Cycle Stages. In: Sharp H., Hall T. (eds) Agile Processes, in Software Engineering, and Extreme Programming. XP 2016. Lecture Notes in Business Information Processing, vol 251. Springer, Cham.