Journal of Cases on Information Technology
Volume 22 • Issue 3 • July-September 2020

Derivation of an Agile Method Construction Set to Optimize the Software Development Process

Jerome Vogel, Swiss Life AG, Switzerland

Rainer Telesko, Fachhochschule Nordwestschweiz, Hochschule für Wirtschaft, Institut für Wirtschaftsinformatik, Switzerland

ABSTRACT

Today, IT and especially the development of customer-focused software, has become one of the most important elements for a company to remain competitive. The primary purpose of this article is to determine whether it is possible to improve the agile development of a company by adjusting its software development process in terms of cultural, technical, and managerial dimensions. Based on a literature analysis and practical experience in the company, different influencing factors and parameters, which affect the value creation throughout the software development process of a company were derived and clustered. Out of these clusters, a framework (Agile Method Construction Set) based on a Microsoft Excel questionnaire was created in order to analyze and identify optimization potential in the development process of a company. This construction set was adapted to meet the requirements of the development department of a large Swiss insurance company in order to validate and test the framework with real data.

KEYWORDS

Agile Best Practice Model, Agile Maturity Models (AMM), Agile Methods, Extreme Programming (XP), Feature Driven Development (FDD), Scaling Agile Models, Scrum, Software Development, Software Process Improvement

INTRODUCTION

Using agile methods in software engineering is a hot topic nowadays. Agile methods are usually the first choice especially when the scope of a task is unknown at the very beginning or requirements are likely to change all the time. However, there are some pitfalls when applying agile methods, which may lead to either success or failure. The main challenges are the following:

- What agile method should be selected? Most projects select Scrum. However, there are also alternatives. Scrum focuses only on management practices. When it comes to prescribing or recommending technical practices as well, it is necessary to enrich Scrum with elements of another agile method.

DOI: 10.4018/JCIT.2020070102

- How can a selected method be tailored in order to create maximum benefit?
- How can this benefit be expressed in terms of Key Performance Indicators (KPIs)?
- How can the agile software development process of a company be optimized?
- Given the fact that an agile method has already been implemented, what are possible optimization sources? What would be a good starting point regarding the maturity of the company? And how could it be measured?

Summarizing the points above in one problem statement one can say: "How can a software development company be supported with regards to identifying a proper agile method, tailoring it and assisting with its continuous improvement?

The primary purpose of this paper is to determine whether it is possible to improve the agile development of a company by adjusting its software development process in terms of cultural, technical and managerial dimensions as well as current development trends.

The paper is organized as follows: First, current well-known approaches (Scrum, Extreme Programming and Feature Driven Development) are briefly presented and their approach-specific specialties are emphasized. Based on a literature analysis there is a strong indication that there are different influencing factors and parameters, which somehow have an impact on the value creation throughout the development process of a company. These different influencing factors and parameters from the literature analysis and additional sources are clustered. Based on these clusters, a framework (the so-called Agile Method Construction Set) based on a Microsoft® Excel questionnaire was created to analyze and identify optimization potential in the development process.

The second part introduces related approaches to Software Process Improvement (SPI). This includes the presentation and brief explanation of the well-known maturity model CMMI® with its process areas and maturity levels. The main challenges of CMMI® are pointed out, such as its generic structure as well as the necessity to produce many (maybe regarding the company context unnecessary) artefacts. The proper way to integrate with agile methods is still ongoing research. The desire to rate and measure maturity has led to the Agile Maturity Models. Although they share many similarities with the Agile Method Construction Set described in this paper, they focus heavily on the Scrum methodology and have therefore many references and dependencies. While most "heavyweight" models scale well and are easy understandable "out of the box", many managers fear that they would lose this scalability when they switch to an agile model. Accordingly, the approaches Scrum of Scrum (SoS) and Scaled Agile Framework show attempts to deal with this.

In the third part, the Agile Method Construction Set was adapted to a development department of a large Swiss insurance company in order to validate and test the framework with real data. The validation was performed with sample KPIs using existing data from the insurance company. The results support the claim that there are general metrics, which affect the development process.

Based on the results, we show that it is possible to improve the agile development of a company by adjusting the software development process in terms of cultural, technical and managerial dimensions as well as current trends. The underlying assumption of this paper is that development processes are regarded as "normal" business processes and can be treated with the whole "arsenal" of modern Business Process Management. Therefore, the focus is on development issues and not on the examination of specific engineering practices.

BACKGROUND

This work belongs to the area of SPI. Although more and more companies are planning or already using agile methods, there is still no silver bullet for the implementation process. Focusing on the wrong priorities and using an inappropriate framework can lead to a lack of acceptance and, ultimately, a failed implementation.

For that purpose, a framework – entitled Agile Method Construction Set – which shows possible entry points for the successful implementation considering a holistic company view was developed and tested. As described in the section about existing approaches, Agile Maturity Models and Scaling Agile Models were a valuable source for identifying important dimensions and key factors of the framework. As opposed to many existing approaches, the Agile Method Construction Set is focusing on details of the technical and organizational implementation of selected agile methodologies and is therefore able to produce fine-grained assessments and recommendations.

COMPARISON OF AGILE METHODS

In the following, the agile methods Scrum, Extreme Programming (XP) and Feature Driven Development (FDD) are briefly presented to ensure a common understanding.

Scrum

The predominant part of agile projects is realized with Scrum (Schwaber & Sutherland, 2016). Scrum is based on the principles of the agile manifest and describes a lightweight procedure model for the development of software in an iterative and incremental way based on business needs. The increments are called Sprints and typically take three to four weeks. Scrum focuses on the management of development and does not prescribe specific development techniques; therefore, it can easily be combined with other approaches, such as XP.

Extreme Programming (XP)

Extreme Programming was created by Kent Beckman based on a project which nearly failed. XP focuses more on concrete engineering practices. These practices, which may be synergistically used together with Scrum, are as follows:

- Pair programming (two programmers are coding/assisting with changing roles);
- Test-driven development (which is maybe the most useful XP practice because it has a profoundly positive effect on system design and quality when it is put into effect);
- Incremental design (including refactoring, works best together with Test-driven development);
- Continuous integration (often difficult to achieve);
- Collective code ownership (difficult to achieve);
- Coding standards (should be done in every modern development team, is the baseline for refactoring);
- Sustainable pace/energized work (productivity and quality will probably improve).

Feature Driven Development (FDD)

Feature Driven Development was created by Jeff De Luca in order to carry out a big and time critical project (DeLuca & Aguanno, 2009). Since then, FDD has continuously evolved. The central element of FDD is the "feature", which should create "value" for the customer.

FDD has many elements that remind of older, more sequential methodologies such as the waterfall model. Its ability to react to changes is therefore limited. But on the other hand, it is well-structured (in five processes) and often chosen in larger-scale projects with limited change of scope. There are still a lot of similarities with Scrum such as the focus on collaboration and communication as well as the development and testing in short iterations. One of the remarkable differences to Scrum is that it offers specific engineering practices (like XP).

AGILE METHOD CONSTRUCTION SET: THE STRUCTURE

Based on the analysis of current best practices, a literature analysis and trends regarding agile development, there is an indication that a variety of different influencing factors and parameters

exist, which somehow have an impact on the value creation throughout the development process of a company. This section gives an overview how the different influencing factors and parameters where identified and clustered. These clusters act as a classification system (e.g. culture is a sub-topic of organization) and build the basis for the creation of the framework (so-called Agile Method Construction Set).

Elicitation of Key Factors

The main sources for eliciting the factors and parameters having an impact on the success of the development process were a post-mortem analysis of conducted projects at Swisslife and an analysis of current research in this field. We analyzed results from both sources and used them as methodological basis for setting up the Agile Method Construction Set.

Considerable research has been carried out in the field to identify factors that have an impact on the success of agile development projects. The following Table 1 gives an overview of the current

Table 1. Compiling important agile success factors from literature

Author	Approach
Shahane, Jamsandekar & Shahane (2014)	Technical factors: Requirements, Development, Testing People factors: Competency, Personal characteristics, Communication and negotiations, Team composition, Societal culture, Training and Learning Organizational factors: Customer commitment, Decision time, Team distribution, Team size, Corporate culture, Planning and control, Safety criticality and reliability, Dynamism and uncertainty
Kouzari, Gerogiannis, Stamelos & Kakarontzas (2015)	Commitment, Staff involvement, Training, Resources, Process Action Teams, Staff experience, Guidance, Reviews and feedback, Implementation Methodology, Monitoring, Communication, Return on Investment, Awareness of SPI
Fatema & Sakib (2017)	Backup behavior, Mutual performance monitoring, Team skills training, Technical training, Working environment, External project factors, Project management, Resource constraints, Task variety and innovation, Goals, Communication, Coordination, Mutual trust, Feedback, Adaptability, Team orientation, Team leadership, Staffing
França, da Silva & Mariz (2010)	Management style, Software process, Team structure, Technology, Team capability, Customer commitment, Delivery strategy, Team location, Customer awareness
Additional project specific factors	Code quality factors (source control, build process, etc.) Software architecture (standardized use of technology, cloud strategy, etc.) Design aspects (usability testing, prototyping, etc.) Development practices (test driven development, pair-programming, etc.) Values (communication, stakeholder mgmt., etc.) Tool support (traceability, standardisation, audit compatibility, etc.)

research in this area. Shahane, Jamsandekar, and Shahane (2014) developed a framework containing important factors from a literature survey, which will serve as a basis for validation in the Indian IT industry. Kouzari, Gerogiannis, Stamelos, and Kakarontzas (2015) focused in particular on critical success factors and barriers in lightweight SPI. Fatema and Sakib (2017) investigate what factors influence the productivity of teamwork in agile software development. All three approaches are complementary in the sense that they all focus on different aspects of agile development (i.e. literature survey, software process improvement, agile teamwork) and therefore give a complete overview (a kind of superset) of all relevant factors. França, da Silva, and Mariz (2010) discuss, based on a cross-

sectional survey, to what extent the adoption of twenty-five agile attributes (clustered into seven components) is related to the success of the Scrum project.

Deriving the Agile Method Construction Set

From the literature analysis and the projects undertaken at Swisslife it became quickly clear that all factors should be clustered around the dimensions organization, people and technology:

- **Organization:** Here the main issue is how an organization needs to be set up in order to guarantee an optimal environment for development. It focuses on the conditions which influence development and its process as a whole:
  - **Coverage/Process Strength:** The subarea coverage/process strength primarily focuses on the organizational elements of the whole value creation process (from a first request until maintenance). This includes different aspects such as traceability, bug tracking, data analytics, stakeholder management and operations;
  - **Culture:** This topic highlights intangible cultural aspects. Corporate values, management support and communication are important aspects. It also addresses the company context, which affects the choice of the underlining agile framework/methodology (e.g. Scrum);
  - **Goal Setting:** The questions within this subarea try to find out whether the development principles are derived from the strategy, mission and vision. It also covers some key aspects of the set goals like customer centricity and data analytics;
  - **Monitoring/KPI:** The most important KPIs a company needs to have as a baseline for development (project costs, work items, reporting standards) are discussed here;
- **People:** This area addresses the skills and roles of the employees. A proper match between roles, skills and requirements is the key to having the right people in the right position:
  - **Skills:** This subarea covers all the elements that assure that the employees have the right skills/method competencies and that they are used in an optimal way (training plan, knowledge management, etc.);
  - **Roles:** This subarea contains questions regarding the clarity of roles. Cleary defined roles are important in order to ensure clarity in terms of responsibility and authority;
- **Technological Environment:** This area focuses on the actual development. It includes the development process as well as the environment and the technology used. This section contains 2/3 of all influencing factors and is therefore the most important one:
  - **Code Quality:** The factors in the code quality subarea are mainly derived from "The Joel Test: 12 Steps to Better Code" (Spolsky, 2000) and other sources (e.g. usage of tools for quality assurance);
  - **Development Environment:** This subarea deals with the development environment. Influencing factors are the technical environment (testing environment, etc.), tool support, security concepts, etc.;
  - **Development Process:** This subarea deals with the development process itself. Therefore, it mainly focuses on the development part of the various agile approaches;
  - **Monitoring/KPI:** This subarea covers the most important KPIs a company needs to have as a baseline for the development process itself (e.g. burn-down charts, bug tracking, KPI integration for new features, etc.).

THE AGILE METHOD CONSTRUCTION SET

In a first step, we derived questions from different influencing factors in order to assess the situation within a company and compiled them in the form of a questionnaire realized with Microsoft® Excel. The Agile Method Construction Set is therefore a standardized, best-practice

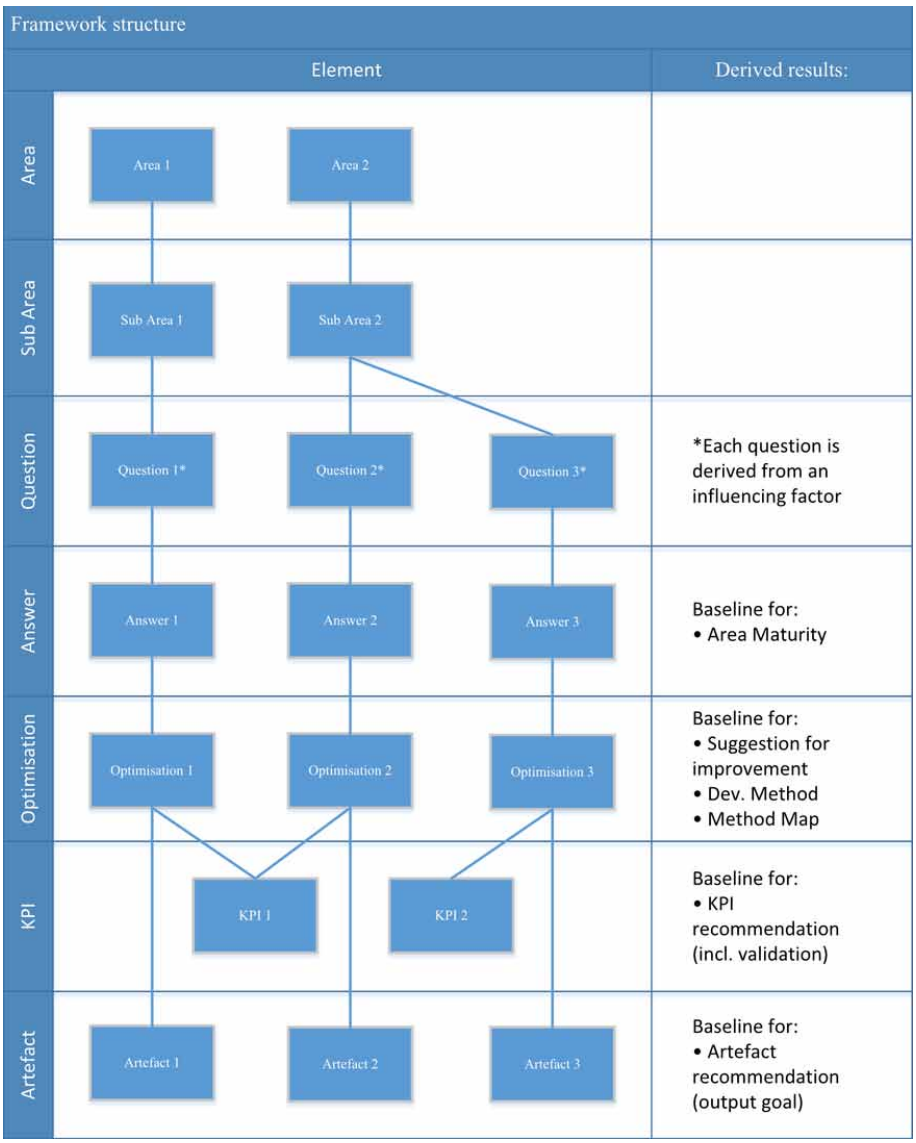
tool to analyze the development process in relation to different viewpoints. To illustrate the framework structure, see Figure 1.

Important elements are now explained in detail. The answers build the baseline for the maturity analysis of the company in relation to the various areas (organization, people and technological environment). According to the answers, we propose suggestions for optimization.

The suggested optimizations build the basis for the recommended agile approach as well as the company-adjusted software development process. For every optimization, affected KPIs and suggested artefacts will be derived. The KPIs build the baseline for the validation of the optimization (Is there really a need for action?) and are required to define the initial situation within the company.

The artefacts help the company to clarify the expected results based on the implementation of the framework. They are also a part of the company's adjusted software development process.

Figure 1. Structure of the agile method construction set



In the following, two examples illustrate the elements of the Agile Method Construction Set. The first example (Table 2) refers to the area "Organization" and in particular to the setting of goals.

The second example (Table 3) refers to the area "Technological Environment" and in particular to the Development Process.

Table 2. Example record from the agile method construction set

Area	Organization
Subarea	Goal Setting
Question	Are results from data analytics used as elements for the goal setting process?
Answer	Data analytics is in use, but the results are not used in a standardized way and are not reflected within the goal plan.
Optimization	Data analytics is an important factor in figuring out what the user really wants and what features are the most effective to develop. This data should be used within the goal setting process.
KPI	Maturity of the organization's analytics initiatives
Artefact	Defined process for data analytics

Table 3. Example record from the agile method construction set

Area	Technology Environment
Subarea	Development process
Question	Is the code quality assured (e.g. via Sonar)?
Answer	The code quality assurance is not performed automatically or on a regular basis, and/or is not based on defined code standards.
Optimization	The code quality assurance is important in order to ensure a certain level of quality within the code and thus achieve reduced maintenance costs.
KPI	Time spent for refactoring, Bugs found in unit tests or UAT resulting from bad code quality, Features delivered on time.
Artefact	Code quality analysis tool

Limitations/Validation Content

The framework focuses mainly on the development process. Preconditions for the application of the framework are:

- The company needs to already have a working operation model;
- The development (or at least its coordination) needs to be onshore. Fully outsourced development teams cannot be properly rated and are therefore out of scope;
- The framework focuses on midsize development teams/departments.

Evaluation Sheets

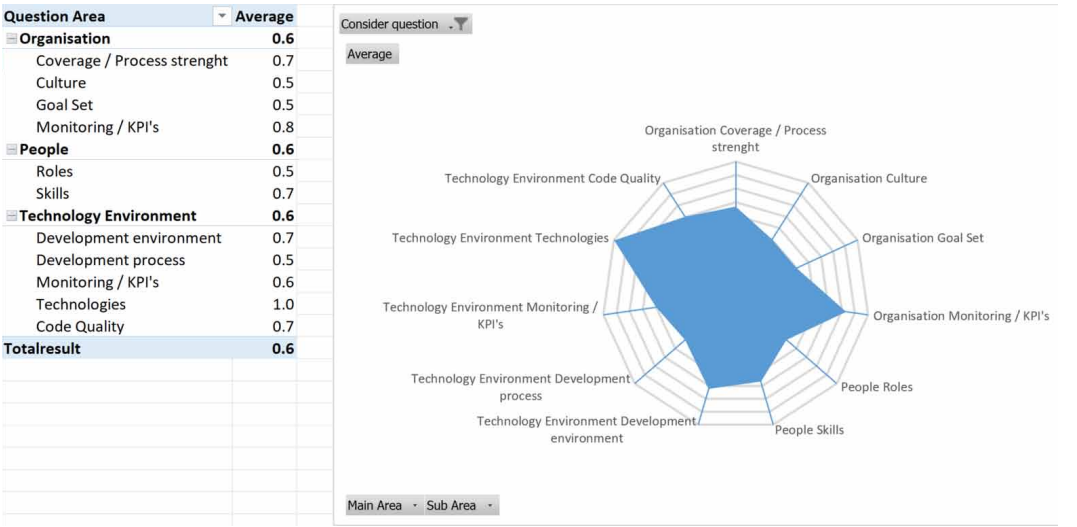
For the various questions, results will be displayed in the form of evaluation sheets. They could be regarded as viewpoints.

Department Maturity

The maturity sheet gives a graphical overview of the company's maturity in the different areas based on the given answers. Like in other maturity frameworks (e.g. CMMI®), this allows a structured rating about the status and improvement potential of a company (Chrissis, Konrad & Shrum, 2011). Best maturity is 1, worst is 0.

Within the calculation sheet, the value can be adjusted based on company-specific focus points. The maturity (see an example in Figure 2) only gives a rough idea regarding the current state. It can be regarded as a starting point for the measurement of the improvement.

Figure 2. Department maturity



Question Area	Average
Organisation	**0.6**
Coverage / Process strenght	0.7
Culture	0.5
Goal Set	0.5
Monitoring / KPI's	0.8
People	**0.6**
Roles	0.5
Skills	0.7
Technology Environment	**0.6**
Development environment	0.7
Development process	0.5
Monitoring / KPI's	0.6
Technologies	1.0
Code Quality	0.7
Totalresult	**0.6**

Suggestion for Improvement

The "suggestion for improvement" sheet compiles the suggestions for improvements in the different areas based on the given answers. This helps the company to find out which areas need attention.

Recommendation Regarding the Development Framework/Methodology

The recommendation helps to locate the best fitting development framework/methodology. It helps to gain insight into why a specific agile approach is more suitable than others (e.g. FDD gets more points in the subarea organization/culture than Scrum, because it works best if there is a stable company environment). It is very important that the development process relies on a standard and is only adapted as far as required to ensure completeness and patency (Schwaber & Aguanno, 2009).

The framework/methodology with the most points is probably the best baseline for the rated company. However, this is only an assumption, which needs to be verified in detail. Because it is impossible to consider all the company specific peculiarities, additional information should be collected in a pilot project before making a final judgement.

Recommendation of "KPIs"

The recommendation of KPIs lists all the KPIs related to the suggested optimization based on the questionnaire. This allows the company to keep track whether the optimization has really brought a benefit and (even more important) to verify in the validation whether there is really a need for

optimization. The use of the right KPIs ensures that everyone is focused on the actual goal, held accountable and that improvements are tracked.

Recommendation "Artefacts"

All artefacts related to the suggested optimization are listed here. This allows the company to see in advance which artefacts will be needed in order to really achieve an optimization. It also allows the company to make their deliverables comparable between the teams and to gather a common understanding, because many artefacts, such as the burndown charts can be used in conjunction with various methods (Mittal, 2013).

Benefits and Disadvantages of the Framework

Since the framework is based on standardized questions leading to a predefined optimization/result based on the answers, the outcome should be less subjective than in an analysis without a standardized procedure.

The generated results only give a starting point in terms of improving the department and do not replace the experience of stakeholders. Sometimes the root cause is deeper, e.g. in a lack of resources due to a lack of budget which leads to a weak development/operation process. Therefore, it is important to deeply investigate the related KPIs before implementing a project.

RELATED APPROACHES

In this chapter, we describe related approaches for our new framework.

Maturity Models

For SPI there exist some well-known industrial frameworks like CMMI® and ISO/IEC 15504. CMMI® (Chrissis, Konrad & Shrum, 2011) is a best practice maturity framework providing versions (so-called constellations) for acquisition, development and service domains. Each constellation consists of a set of predefined process areas (i.e. processes) containing specific practices which describe their requirements. A company is compliant to CMMI® if it succeeds in implementing the necessary specific and generic practices (which are applicable for all process areas). Five maturity levels (ML) are distinguished for all constellations:

- **ML1 – Initial:** No adherence to CMMI® processes is evident ("chaos");
- **ML2 – Managed:** Basic project management is in place;
- **ML3 – Defined:** Standard processes for project management and engineering are in place and can be tailored according to the needs of the projects undertaken;
- **ML4 – Quantitatively Managed:** Processes can be managed based on predefined metrics;
- **ML5 – Optimizing:** Processes can be optimized based on the insights of ML4.

CMMI® is interesting for our paper for several reasons. First, it elaborates a framework (CMMI®-DEV – CMMI® for Development) to measure the maturity of development organizations. Secondly, it defines software engineering practices like Verification, Validation, Product Integration, etc. Because CMMI® does not prescribe any procedure model it can be applied not only to Scrum, but also to the Waterfall model, V-Model, Rational Unified Process, etc. Therefore, the description is generic to a certain extent and needs to be tailored to meet the specific requirements of a development environment and to be truly useful. Finally, CMMI® also provides a built-in improvement cycle called IDEAL℠ consisting of the following phases (Kautz, Hansen & Thaysen, 2000):

- **Initiating:** Establishing an improvement infrastructure and defining the goals of the initiative;

- **Diagnosing:** Identifying the organization's initial maturity via an appraisal;
- **Establishing:** Developing an action plan draft with measurable goals;
- **Acting:** The implementation of the solutions that handle the identified weaknesses is created, piloted, and deployed;
- **Leveraging:** Collection of lessons learned to be applied in subsequent IDEAL℠ rounds.

Similar to the process areas, the IDEAL℠ method needs to be tailored to meet the requirements of a development organization. Such a tailoring approach to IDEAL℠ for a small enterprise is shown by Kautz, Hansen and Thaysen (2000).

The way to properly integrate CMMI® with agile methods is still being explored. CMMI® is a so-called "heavyweight" model and demands the creation of many "artefacts" (i.e. documents such as requirements specifications, test reports, meeting minutes, process flow charts, etc.) in the respective process areas. Agile methods, on the other hand, claim that knowledge exchange should be done mainly via meetings with the consequence to minimize the documentation. To resolve this conflict and at the same time adhere to both CMMI® and agile principles is obviously a true challenge and also depends on the maturity of the development organization. A general overview of the combination of CMMI® and agile approaches can be found in Lukasiewicz and Miler (2012). An approach to implement ML2 in an agile organization (Web Development) is described by Torrecilla-Salinas, Escalona and Mejías (2012). The right combination of CMMI® and Scrum, especially in small and medium enterprises is shown by Lina and Dan (2012).

Overall, CMMI® is a well-established maturity framework with many existing implementations, especially in large organizations. Because of its heavyweight nature and thus obvious conflicts with agile principles, the effort of a synergistic integration with e.g. Scrum is relatively high.

Agile Maturity Models (AMM)

Adapting a maturity framework such as CMMI® for use in an agile environment (typically Scrum) leads to Agile Maturity Models (AMM). Most of the models have the typical 5-level structure (Ambler, 2010). A good overview and an assessment of the approaches is given by Buglione (2011). Schweigert, Vohwinkel, Korsaa, Nevalainen, and Biro (2014) discuss structure, quality and content of the currently available agile maturity models.

A non-standard approach – in the sense of adopting a CMMI®-like model for an AMM – can be found in (Gani & Starrett, 2017). This approach is highly interesting because it has some similarities with the model components (influence factors and maturity assessment) of our Agile Method Construction Set.

The basis for the approach of Gani and Starrett is the so-called "Organizational Ecosystem Model" with four pillars (Starrett, 2014):

- **Organizational Dynamics:** This is where cultural factors come into play such as history, traditions, norms, relationships, spheres of influence, and more;
- **Product Management:** This pillar deals with vision, attention, guidance and focus, all factors that are important for successful product development teams;
- **Agile Principles and Scrum Practices:** This pillar focuses on the Scrum practices and to what extent Scrum practices such as setting up a product/sprint backlog, review, retrospective, etc. have been internalized;
- **Technical Management:** Important factors here include strong designs, UI, architecture, creative solutions, and competitive advantage, etc.

Agile maturity is measured by rating five factors in each pillar, which makes twenty factors as a whole. For scoring each factor, the so-called "Shu Ha Ri" model is used. It consists of the three stages of maturity Shu, Ha and Ri which have the following meaning (Gani & Starrett, 2017):

- **Shu (value 1):** lowest maturity, follow the rule (learning);
- **Ha (value 2):** intermediate maturity, adapt the rule (detachment);
- **Ri (value 3):** highest maturity, become the rule (optimization).

Based on the maturity of each factor, the strengths and weaknesses for the implementation of each pillar can be depicted in a spider diagram, which is very similar to our approach shown in Figure 2. Finally, an overall maturity results from considering all pillars. This enterprise maturity based on the "Organizational Ecosystem Model" is comparable to the maturity in CMMI®, however, the model of Gani and Starrett only considers companies that aim to properly implement the agile method Scum in their development departments or teams.

Scaling Agile Models

A challenge that is specific to all agile methods is scalability. Heavyweight procedure models like the Waterfall model or Rational Unified Process usually scale well, because they use documentation (artefacts) as a central medium for information exchange. Agile methods demand that the documentation should be reduced to a minimum wherever possible and that the exchange of knowledge has to be promoted via meetings such as a sprint planning meeting, reviews, etc.

The main disadvantage of this approach is that meetings may be effective if five to ten participants attend them. In a larger project with fifty attendees, the logistical effort and synchronization of the team members drastically increases the risk for project failure. In order to minimize such risks, specific methods have been developed to support scalability (Ebert & Paasivaara, 2017).

Scrum of Scrums (SoS)

SoS is a lightweight approach for scaling Scrum. When the number of team members exceeds a certain threshold (e.g. very often seven to ten members), two or more teams are created to manage the tasks in the Sprints. In order to coordinate the work (i.e. analyzing, implementing and testing features) synchronization between the teams is required. This is managed by setting up a Scrum of Scrums meeting (SoS) where dedicated team members of the respective Scrum teams participate to support synchronization. If the development project is very large and challenging and, for example, includes multiple subsidiaries this approach can be very easily extended in a cascading manner. SoS is an easy way to scale Scrum at the Enterprise level, following the basic principles of agile development and not overloading the organization. However, the basic requirements for succeeding with SoS are a high maturity level for Scrum and experienced team members, especially those responsible for product vision and software architecture.

Scaled Agile Framework (SAFe®)

SAFe® is an industry-proved approach to scale agile methods to the Enterprise level and demands much more effort than SoS. The latest SAFe® version 4 (Full SAFe®) distinguishes four levels which are briefly presented here starting with the smallest and ending with the largest level (Scaled Agile Inc., 2017):

- **Team:** This (smallest) level contains the usual practices known from the original definition of Scrum, such as setting up a Product Backlog, a Sprint Backlog, performing a review, etc. This work is performed in the context of the so-called Agile Release Train (ART);
- **Program:** This level describes the work of the ARTs, a group of eight to twelve teams, which are funded to achieve long-term development based on the enterprise strategy;
- **Large solution:** This level contains the roles, artifacts, and processes needed to build large solutions and differs from the previous level by focusing more strongly on the coordination of

the various ARTs incl. suppliers, managing requirements engineering and mastering compliance and regulations;

- **Portfolio:** The portfolio management level of SAFe® focuses on the entire system and "contains the principles, practices, and roles needed to initiate and govern a set of development Value Streams" (Scaled Agile Inc., 2017). A Value Stream (typically cross-functional, -organizational, and -geographical), belongs – besides ART – to the organizational backbone of SAFe® 4. It describes, very similar to a business process, a series of development steps that are initiated by a trigger to provide values for the end customer.

SAFe® has been successfully implemented in large companies. Paasivaara (2017) describes the adoption of SAFe® in two business lines at Comptel, a globally distributed software development company.

Assessing Related Approaches for Developing the Framework

This paper deals with the derivation of an Agile Method Construction Set in order to improve the software development process. Related approaches are particularly concerned with the maturity and scalability of development organizations:

- **Maturity:** In the further development of the Agile Method Construction Set, it has to be checked whether it makes sense to adopt selected concepts of CMMI® or AMMs. The four pillars of the "Organizational Ecosystem Model" (Starrett, 2014), which are a valuable source for the next release of the Agile Method Construction Set, are particularly interesting here;
- **Scalability:** Scalability could be considered as a new subarea within the Agile Method Construction Set. Elements of SoS or SAFe®, e.g. the organizational capability to set up ARTs could be adopted in order to check whether an organization is prepared to properly scale Scrum.

DESCRIPTION OF THE PILOT AND EVALUATION OF THE AGILE METHOD CONSTRUCTION SET

To validate the framework, a development department was chosen in one of the leading Swiss insurance companies. The specific KPIs have been validated with data from past projects of the selected department. This chapter gives a short overview of the department structure and assessment procedure. Because this is only intended to illustrate the process, only sample data is used.

Department Structure

To validate the framework, a development department was chosen in one of the leading Swiss insurance companies. The department is relatively new (less than three years old) and its task is to "perform strategic projects within the insurance sector related to digitalization". It consists of over 60 employees, of whom 40 are developers. They are divided into four teams with 6 to 14 employees.

General Assessment Procedure

Initially, the various stakeholders were identified (group of developers, head of department, release manager, Scrum Master, system architect and tester). Meetings were scheduled and the questionnaire was filled in. The framework resulted in:

- An auto-generated list of suggestions for improvement
- KPIs
- Recommended artefacts
- Recommendation for a baseline methodology/framework

Before the changes are implemented, the relevant KPIs for the defined areas of action have to be validated. This defines a starting point for the measurement of the improvement and ensures that there is a real need for optimization. The questionnaire is always answered based on the perspective of an individual stakeholder. Since every participant answers the questionnaire from his own point of view, there is a risk that this will end with contradictions. For example, a stakeholder feels a lack of management support and answers the question accordingly, but a different stakeholder feels well supported. Thus, there would be a deviation or contradiction, blurring the need for optimization. In the end, the management has to take the final decision on how to structure the new software development process and what suggestions should be implemented and with what priority.

Example Results

Figure 3 shows suggestions for the software process improvement, here recommended artefacts according to the predefined dimensions.

Figure 4 depicts the recommendation of the framework regarding the proposed agile method (here exemplified for Scrum). A detailed scoring for all areas and subareas is provided.

Figure 3. Suggestions for software process improvement

Artefacts
Organisation
Coverage / Process strenght
Dev-Ops based concept for IT operation
Decision support systems accessible for management. Increasing number of decisions taken based on data analytics (results follow up survey)
Stakeholder communication plan, reduced number of misunderstanding between business and it (results follow up survey)
Culture
Defined actions regarding improvement of the management support for the employees. Increasing satisfaction of the employees regarding management support (results follow up survey)
Defined communication-standards, Increasing satisfaction regarding corporate communication (results follow up survey)
Defined and accessible corporate values (compliant with agile manifesto, accessible and written down)
Goal Set
Documentation of roles, goals and responsibilities, Increasing Customer (Business) satisfaction with the development result of the IT (results follow up survey)
Defined process to assure Alignment between goal set out of requirements and vision / strategy
Defined process to assure, that results out of data analytics are used a standardized way and are reflected within the goal plan, increased maturity assessment results
Monitoring / KPI's
A process is in place in order to assure, that business tracks the work items
People
Roles
Documentation of roles, goals and responsibilities
Skills
Standards regarding how skills are used as an element of the team composition
Platform with known skillset of company
Training plan regarding development methodology, increased level of experience regarding refactoring (results follow up survey)
Development plans, capacity planning

Figure 4. Recommended development framework/methodology for the insurance company

Recommendation development method	Sum of Scrum
Organisation	3
Coverage / Process strenght	1
Is the business process regarding new requirements well structured and there are not many scope changes?	1
Scrum and extreme programming work better in an environment with many scope changes then FDD	1
Culture	2
Are the requests of the business rather repetitive than project oriented?	1
With it's feature orientation and the short feedback loops Scrum and extreme programming are better in handling projects then FDD. Extreme programming even allows changes within the iterations.	1
Is the company environment quite stable and not subject of a high number of significant changes e.g. throughout new technologies?	1
Scrum and extreme programming work better in an environment with changing conditions then FDD	1
Technology Environment	6
Development process	6
Are there many scope changes which affects the development team within a short time (below 1 month)	0
Extreme Programming allows change within an iteration. As long as the team hasn't started work on a particular feature, a new feature of equivalent size can be swapped into the XP team's iteration in exchange for the not yet started feature	0
Are there mini specification defined and implemented within the unit tests? (TDD)	1
Neither Scrum nor FDD specifies or requires the implementation of mini specifications	1
Is collective code ownership used within the development?	0
FDD uses class ownership (code assigned to a single owner)	0
Is continuous integration a central goal for the development?	1
Scrum an FDD doesn't know continuous integration, resp. continuous integration is intercepted by the definition of done. Anyway, continuous integration has many benefits like faster build (which in the end means faster go to market), decreased code rev	1
Is refactoring a specified and central element of the development framework within the company?	0
Extreme programming provides guidelines regarding refactoring	0
Is there a (it) role which represents the development team against the customer and defends project results over stakeholders, supervisors (e.g. a Project Manager)	2
At Scrum, the development team itself overtakes the role of the IT-project manager	2
Is there a role which tracks the status of the programmers and compares them with the estimated target time (e.g. a schedule manager)	2
FDD and Scrum doesn't know the dedicated role the schedule manager. The responsibilities are split among the team.	2
Score	9

CONCLUSION

Today, the development process of an enterprise is highly dependent on the internal and external environment of a company. It needs to be ensured that the company can react quickly to changing market conditions while adopting the role of an enabler.

The evaluation of the available literature and the different agile approaches have shown that although there are different agile frameworks/methodologies, none of them can be used as a "silver bullet" for every company or scenario. The implementation depends to a large extent on the company-specific setup and environment. Elements of different agile frameworks can also be combined as long as the underlying nature of a framework remains untouched (e.g. Sprints in Scrum).

Based on a literature analysis about agile development, there is a strong indication that there are various factors and parameters, which influence the value creation throughout the agile development process of a company.

A framework (Agile Method Construction Set) was created, based on Microsoft® Excel, bundling all influencing factors in the form of a questionnaire. This framework was tested within a development division of a large Swiss insurance company.

The results from the cross validation support the claim that there are general metrics, which affect the development process. An analysis of the identified KPIs (based on historical company data) emphasized the feasibility of the approach. Further investigation showed that it is very likely that the application of the suggested optimizations will have a positive effect in the future. This was further underlined by the statements of the head of department, and senior software engineers of the department. Their overall feedback indicated that the proposed redesign of the software development process is comprehensible and applicable. An adaptation - as recommended by the Agile Method Construction Set - would have a positive effect on the value creation in the software development process and thus on the whole company.

Concerning future research, we see high potential for the integration of the Agile Method Construction Set and existing approaches. Finally, two possible approaches are outlined:

- Successful scalability management is an important issue for large companies. When identifying weaknesses using the Agile Method Construction Set, a higher degree of maturity can be achieved in this specific topic by applying e.g. SAFE® or SoS. This implies that the Agile Method Construction Set comprises a set of methods, which can be dynamically adapted and expanded according to the latest developments in industry and research;
- Another research direction could be to focus more on empirical results. It would be interesting to select different cases from industry and to perform assessments with the maturity models. One could gain valuable insights by comparing the results with an in-depth analysis of key stakeholders such as project managers and customer representatives ("Gap analysis"). Such a procedure would make it possible to identify the fundamental strengths and weaknesses of the models in terms of their structure (i.e. on what basis is the assessment performed?) and their rating algorithms.

REFERENCES

Ambler, S. (2010). *The Agile Maturity Model (AMM)*. Dr. Dobbs. Retrieved from http://drdobbs.com/architectureand-design/224201005

Buglione, L. (2011). Light maturity models (LMM): An Agile application. In *Proceedings of the 12th International Conference on Product Focused Software Development and Process Improvement* (pp. 57-61). Academic Press. doi:10.1145/2181101.2181115

Chrissis, M. B., Konrad, M., & Shrum, S. (2011). *CMMI for Development: Guidelines for Process Integration and Product Improvement* (3rd ed.). Boston, MA: Addison Wesley.

DeLuca, J., & Aguanno, K. (2009). *The Story Behind Feature-Driven Development. Audio CD*. Multi-Media Publications Inc.

Ebert, C., & Paasivaara, M. (2017). Scaling Agile. *IEEE Software*, *34*(6), 98–103. doi:10.1109/MS.2017.4121226

Fatema, I., & Sakib, K. (2017). Factors Influencing Productivity of Agile Software Development Teamwork: A Qualitative System Dynamics Approach. In *Proceedings of the 24th Asia-Pacific Software Engineering Conference* (pp. 737-742). Academic Press. doi:10.1109/APSEC.2017.95

França, A., da Silva, F., & Mariz, L. (2010). An Empirical Study on the Relationship between the Use of Agile Practices and the Success of Software Projects that Use Scrum. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (pp. 37.1-37.4). Academic Press.

Gani, A., & Starrett, J. (2017). *Measuring Agile Maturity in the Enterprise*. Scrum Alliance. Retrieved from https://www.scrumalliance.org/community/articles/2017/april/measuring-agile-maturity-in-the-enterprise

Kautz, K., Hansen, H. W., & Thaysen, K. (2000). Applying and Adjusting a software process improvement model in practice: the use of the ideal model in a small software enterprise. In *Proceedings of the 2000 International Conference on Software Engineering* (pp. 626-633). Academic Press. doi:10.1145/337180.337492

Kouzari, E., Gerogiannis, V. C., Stamelos, I., & Kakarontzas, G. (2015). Critical success factors and barriers for lightweight software process improvement in agile development: A literature review. In *Proceedings of the 10th International Joint Conference on Software Technologies* (pp. 1-9). Academic Press. doi:10.5220/0005555401510159

Lina, Z., & Dan, S. (2012). Research on Combining Scrum with CMMI in Small and Medium Organizations, In *Proceedings of the International Conference on Computer Science and Electronics Engineering* (pp. 554-557). Academic Press. doi:10.1109/ICCSEE.2012.477

Lukasiewicz, K., & Miler, J. (2012). Improving agility and discipline of software development with the Scrum and CMMI. *IET Software*, *6*(5), 416–422. doi:10.1049/iet-sen.2011.0193

Mittal, N. (2013). The Burn-Down Chart: An Effective Planning and Tracking Tool. Scrum Alliance. Retrieved from https://www.scrumalliance.org/community/member-articles/436

Paasivaara, M. (2017). Adopting SAFe to Scale Agile in a Globally Distributed Organization. In *Proceedings of the 2017 IEEE 12th International Conference on Global Software Engineering* (pp. 36-40). IEEE Press. doi:10.1109/ICGSE.2017.15

Scaled Agile Inc. (2017). *Scaled Agile Framework – SAFe for Lean Enterprises*. Retrieved from http://www.scaledagileframework.com/

Schwaber, K., & Aguanno, K. (2009). *ScrumButs: The Dangers of Customizing Scrum [Audio CD]*. Multi-Media Publications Inc.

Schwaber, K., & Sutherland, J. (2016). *The Scrum Guide: The Definitive Guide to Scrum*. Retrieved from http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf

Schweigert, T., Vohwinkel, D., Korsaa, M., Nevalainen, R., & Biro, M. (2014). Agile maturity model: Analyzing agile maturity characteristics from the SPICE perspective. *Journal of Software: Evolution and Process*, *26*(5), 513–520.

Shahane, D., Jamsandekar, P., & Shahane, D. (2014). Factors influencing the agile methods in practice - Literature survey & review. In *Proceedings of the 2014 International Conference on Computing for Sustainable Global Development* (pp. 556-560). Academic Press. doi:10.1109/IndiaCom.2014.6828020

Spolsky, J. (2000). *The Joel Test: 12 Steps to Better Code*. Joel on software. Retrieved from https://www.joelonsoftware.com/2000/08/09/the-joel-test-12-steps-to-better-code/

Starrett, J. (2014). *Agile and the Organizational Ecosystem*. Scrum Alliance. Retrieved from https://www.scrumalliance.org/community/articles/2014/august/agile-and-the-organizational-ecosystem

Torrecilla-Salinas, C. J., Escalona, M. J., & Mejías, M. (2012). A Scrum-based Approach to CMMI Maturity Level 2 in Web Development Environments. In *Proceedings of the 14th International Conference on Information Integration and Web-based Applications & Services* (pp. 282-285). Academic Press.

Jerome Vogel received a Master of Science in Business Information Systems from Fachhochschule Nordwestschweiz and is currently responsible for the agile transformation of the portal landscape at Swiss Life.

Rainer Telesko is a Professor for Business Information Systems at the University of Applied Sciences Northwestern Switzerland since 2007. He received a MSc and PhD in Business Information Science from the University of Vienna. His research fields and interests are software and requirements engineering, programming, and business process management.