COMPARATIVE ANALYSIS OF AGILE METHODS FOR MANAGING SOFTWARE PROJECTS

Petar Bogojević
Saga d.o.o., New Frontier Group, Belgrade, Serbia

Abstract: The purpose of this paper is to review and compare four of the widely used and referenced agile methods – Spiral model, Dynamic System Development Method, Scrum, and Extreme programming. These four methods are compared based on their process, roles, current research, project management, lifecycle coverage and practices. The result of this paper is a review and comparison of these four models, which shows that neither of the described methods provides full product life-cycle coverage. XP is concluded to be most specific when it comes to practical guidelines, but with a very limited scope. Other methods focus more on abstract principles. Spiral, DSDM and Scrum can be used as frameworks that can use other agile methods. Organizations should use principals and ideas behind these fours methods as inspiration whencreating custom tailored development processes. This paper also provides a review of the current research on these four methods, therefore it can be used as a reference work for future studies.

Key words: Software developments, Agile methods, Spiral model, Scrum, Extreme programming, DSDM,Comparison

1. INTRODUCTION

In the last 40 years, the field of software development has, in its path to become more productive, evolved in the form of developing new methodologies.Software development methods attempted to offer an answer to an eager business community asking for a faster and nimbler software development process (Abrahamsson, Warsta, Siponen, & Ronkainen, 2003). Since the 2000's, this developing path and the way software is being built has been named "agile". The Manifesto for Agile Software Development, written in 2001 by a group of software developers, has given a new momentum in thislightweight way of developing. However, the Manifesto is solely responsible for naming such an approach. Agile practices were being used by companies and practitioners since the 70's (Larman & Basili, 2003). Ever since the Manifesto was written, researchers have been trying to explicate agility and its different facets. In essence, as evidenced in (Dingsøyr, Nerur, Balijepally, & Moe, 2012)agile ideas suggest a "light' methodology that promotes maneuverability and speed of response. (Highsmith & Cockburn, 2001)mention that agile methods did not bring new practices, but recognized people as the primary drivers of project success: "… (agile approach) gives a new combination of values and principles that define an *agile* world view."

Figure 1 presents an evolutionary path of agile software development methods and their relationships. This figure also indicates (using a dashed line), which approaches influenced the making of the Manifesto for Agile Software Development. Figure 1 covers more than the scope of this paper, however in combination with work from (Larman & Basili, 2003), it can demonstrate the entire evolutionary roadbehind agile software development methods.

Corresponding author. Email: petar.bogojevic@gmail.com

ISSN 2560-4961(online)

www.epmj.org

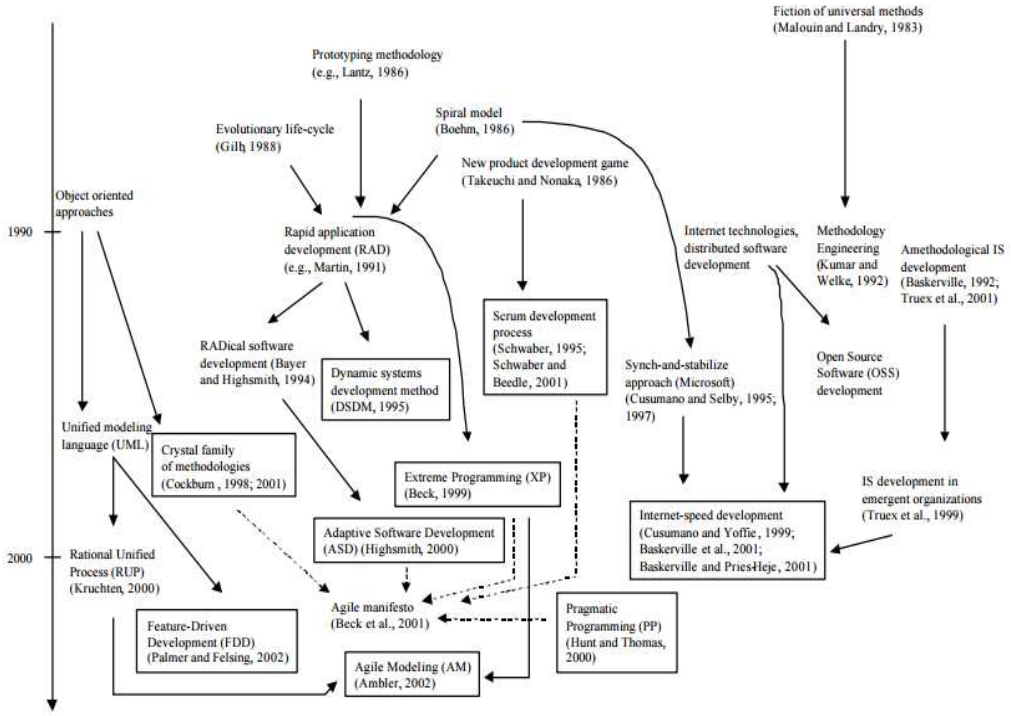

Figure 1: The evolutionary road of agile methods
(Abrahamsson, Warsta, Siponen, & Ronkainen, 2003)

(Dingsøyr, Nerur, Balijepally, & Moe, 2012)studied the period from the creation of the Manifesto for Agile Software Development until 2012, and analyzed papers and conference proceedings written about agile software development. Same authors concluded that ever since 2001 and especially 2005 there has been an increase in journal articles which is a sign of an increase in quality as well as quantity.

(Dingsøyr, Nerur, Balijepally, & Moe, 2012)summarize the majority of definitions from some of the most recognizable papers written in the last decade, regarding agile software development. (Abrahamsson, Salo, Ronkainen, & Warsta, 2002) reviewed the knowledge base of agile software development and introduced a definition of agile development: "Agile software development is incremental (small software releases, with rapid cycles), cooperative (customer and developers working constantly together with close communication), straightforward (the method itself is easy to learn and to modify, well documented) and adaptive (able to make last moment changes)." The authors also state that the development team works by concentrating only on the functions needed at first hand, delivering them fast, by collecting feedback for the work, and by reacting to received information.Generally speaking, according to the (Dingsøyr, Nerur, Balijepally, & Moe, 2012) all of the so called agile methods strived to address the core values of the Manifesto. The values behind Manifesto's four main principles: (1) Distinctive move towards collaborative development – people-centric; (2) Adoption of a "lean" mentality, meaning a need to minimize unnecessary work, with regards to creating unnecessary documentation; (3) Customers and stakeholder are actively involved in the evolution of software development; (4) Acceptance of the fact that software development is an unpredictable and changing process, are summarized by Dingsøyr, *et al.*.

This paper is composed as follows. The subsequent section reviews four agile methods. Following is the comparison of these four methods. The final section provides a conclusion and a brief description of future work.

2. AGILE METHODS REVIEW

Having the right method in place that is well designed and appropriate to the nature of the project and is also well integrated with the customer for the project so that the customer is fully engaged collaboratively in the process is one of key factors of successful project management as evidenced by(Cobb, 2015).

From the examination of the relevant literature on software development, and particularly Figure 1 and the work of (Larman & Basili, 2003),it can be concluded that some methods had, and continue to have influence on the way software is being produced today. On this idea, and with reference to systematic reviews on agile software development,(Abrahamsson, Salo, Ronkainen, & Warsta, 2002; Dyba & Dingsoyr, 2008),a list of four most influenced agile methods was selected for this paper. These are:

- Spiral model
- Dynamic System Development Method
- Scrum
- Extreme programming

The methods are then described based on the model found in (Abrahamsson, Salo, Ronkainen, & Warsta, 2002) and renewed with state of the art literature. Every method is illustrated through the following structure: (1) process – description of phases in the product life cycle; (2) Roles and responsibilities – specific roles used in the model; and (3) Current research– overview of the scientific and practical status of the method.

The goals of this research paperare to: (1)describe the principles and mechanisms of the above software development methods; (2) describe the relevant use case experience from various case studies and reviews; and (3) compare these four methods.

2.1. The Spiral model

The Spiral model was introduced in 1986 in a work by Barry Boehm titled“A Spiral Model of Software Development and Enhancement”(Boehm B. W., 1986). One of the best ideas, which came out of the Spiral model, was that development teams should choose a software development process, which defined: the frequency of the increments, the development tool, the details of the plan, and the risk analysis based on different parts or components of the product. This process presented a next step in the evolution of the software development models of the time and was different from the others since these used the above processes in regard to the final product, as a whole.

2.1.1. Process

The Spiral model was used in the definition and development of the Thompson Ramo Wooldridge Software Productivity System (TRW-SPS). The goal of this project was to define an engineering environment which would significantly increase TRW’s software productivity.

Product development process is performed in spirals or rounds. The Spiral model may appear a bit complex at first, especially if you look at the Spiral model graph (Figure 2), but once you understand the logic behind it, the model becomes much more understandable. Actually, in particular situations the spiral model becomes equivalent to one of the existing process models, for example Waterfall. In other situations, it represents a mix of approaches. Again, the selection of a particular process depends on the risks associated with specific parts of the project.

The spiral approach is very much risk-driven and puts risk identification as its primary starting point. The emphasis is on identifying all types of objectives and constraints during each round of the spiral. By doing so, the Spiral model incorporates software quality objectives into software development. Barry’s work suggests using the spiral approach on hardware projects as well. The Spiral model suggests using prototypes, as a risk reduction option, at any development stage. Plans and specifications should be created in such detail that the lack of it does not jeopardize the project.

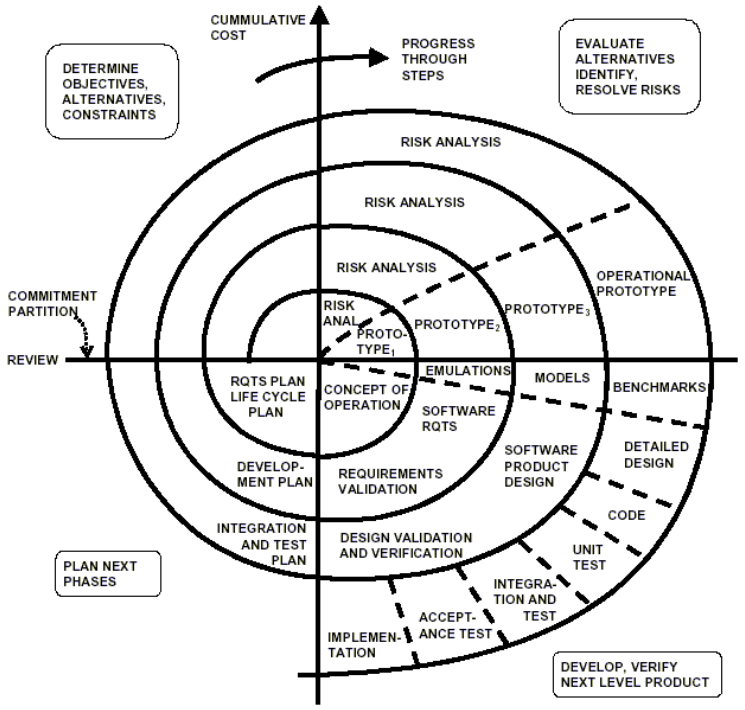

Figure 2: Spiral model of the software process (Boehm B. W., 1986)

2.1.2. Roles and responsibilities

Different from other methods described in this paper, the Spiral method does not define any particular roles. However, it can be inferred from the Spiral process that some roles and responsibilities are necessary for it to work. Besides from the usual roles involved in every software project, such as developers, management and customer representative or business analyst, the main genuine role which appears in the Spiral model is the role of a risk analysis expert.

Risk analysis experts represent a key role in the Spiral model. The Spiral model requires a risk analysis to be performed in every stage of the project. It is also one of the most important parts of this model, since the goal is to pick an alternative based on its risks and risk resolutions. From this, it can be concluded that a person performing the role of a risk analysis expert, must be very familiar with the technology in question, organizational culture, its processes and various alternatives to achieve the goal of the project.

2.1.3. Current research

(Hendrix & Schneider, 2002)present how Boehm's Spiral model was used in NASA's TReKproject. The purpose of this project was to create a PC based ground system which would enable scientist to monitor and control experiments located onboard the International Space Station. It was important for the TReK project to address all the fundamental issues associated with a software development project, including documentation, configuration management, testing, and other quality assurance activities. Like most other software projects, this project faced many challenges such as the need for early product availability for customers, limited personnel resources, a tight schedule, and dependencies on external systems, many of which were to be developed in parallel with TReK by different development teams.

The TReKsoftware lifecycle used the Spiral model in such a way that it was possible to combine both project management and software development. Hendrix *et al.* claim that most of the challenges faced on the TReK project were overcome with support of the Spiral project lifecycle.

Three main disadvantages of the Spiral modelare: (1) it is highly dependable on risk analysis, and risk-assessment experts – the Spiral model suggest that software developers

do the risk analysis, which may prove troublesome in situations where these resources are working on multiple projects or when there is a lack of senior developers to review these risk documents; (2) it is not appropriately adjusted to work on contract software projects – theSpiral model was created and suited to fit the TRW's internal development processes; (3) it needs further elaboration and defining in areas such as contracting, specifications, milestones, reviews, scheduling, status monitoring, and risk area identification to be fully useable in all situations, as stated by (Boehm B. W., 1986).

In his later works, (Boehm &Belz, 1989;Boehm & Turner, 2015), Boehm*et al.* elaborated his use of the Spiral model as a process model generator. As described: "A process model generator is a technique which operates on a project's process drivers as inputs to produce a process model for the project which is tailored to its particular process drivers." In theory, the Spiral could be used to define various key project factors, and based on these factors the spiral will "transform" to an another development method, such as waterfall, for example. More practical studies and use cases are missing in order to provide a more detailed review.

Wider implementation of the Spiral model may still not be possible due to current perceiving on risk management. (Dragan, Marija, & Zorica, 2013)analyzed how risk management influences project performance and presented mixed results. Dragan *et al.* state that "Most software developers and project managers perceive risk management processes and activities as extra work and expense." Also, they indicate that companies did not benefit greatly due to risk management techniques. However, authors also mention that assigning a role of risk manager increases the chance of project success. Dragan *et al.* reviewed the results of risk management studies and stated that risk management efforts were small in small projects ($0.1 million – $1 million) and progressed as projects became bigger. This finding can be seen as relative since small companies may find $1 million projects as key enterprise projects and invest a great deal of risk control in them. The results in this study can be remarked as an additional dilemma, when implementing a risk driven approach, such as Spiral.

2.2. Dynamic software development method (DSDM)

DSDM is a method developed by a dedicated consortium in the UK. The first release of this method was in 1994. The fundamental idea behind DSDM is that instead of fixing the amount of functionality to be delivered and then adjust time and resources needed to reach this functionality, it is preferred to fix time and resources and then adjust the amount of functionality accordingly, stated in (Abrahamsson, Warsta, Siponen, & Ronkainen, 2003). Most of what is written in papers and studies about DSDM comes from (Jennifer & Peter, 1997), and handbooks, (DSDM Consortium, 2014) and (DSDM Consortium, 2008), from the DSDM consortium, which promotes DSDM. Some authors mark DSDM as the first truly agile software development method (Abrahamsson, Warsta, Siponen, & Ronkainen, 2003).

2.2.1. Process

DSDM consists of five phases: (1) feasibility study, (2) business study, (3) functional model iteration, (4) design and build iteration, and (5) implementation asdefined by (Abrahamsson, Salo, Ronkainen, & Warsta, 2002). These five phases are presented in figure 3. First two phases are sequential, and done only once. Development work is done in the last three phases, which are iterative and incremental. DSDM approaches iteration through timeboxes. The time allowed for each iteration, including planned results,is planned through timeboxes in advance. A typical timeboxlength is from a few days to a few weeks. In the following section, DSDM's phases are described according to Abrahamsson, *et al.*

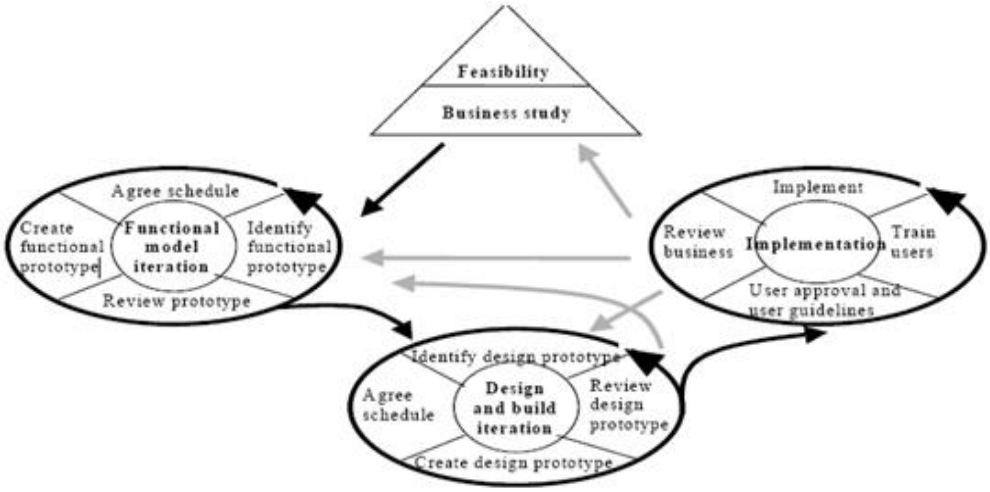

Figure 3: DSDM Process Diagram (Jennifer & Peter, 1997)

In the feasibility study phase, the suitability of applying DSDM, for the given project, is assessed. The feasibility is conducted based on the type of the project and organizational and people issues. In addition, technical possibilities and project risks are analyzed. The results of the feasibility study phase are a feasibility report and an outline plan for development. Optionally, if the team is not familiar enough with the project's business or technology, a fast prototype can also be made. Based on this prototype, the team can make a decision whether to proceed to the next phase or not. Average duration of the feasibility study phase is not expected to take more than a few weeks.

The business study phase is where the essential characteristics of the business and technology are analyzed. DSDM recommends teams to organize workshops,during which a sufficient number of the customer's experts are gathered. This way,the project team can consider all relevant factors of the system and agree on development priorities.The agreed upon business processes and user cases are described in a business area definition. The identification of user cases helps in involving the customer, as key people in the customer's organization areidentified and involved at an early stage. In the business study phase, high level descriptions of the processes are presented, in a suitable format (diagrams, business object models, etc.). The result of the business study phase is a system architecture definition, which sets the first system architecture sketch, and is allowed to change during the project. Also an outline prototyping plan is formed, which states the prototyping strategy for the following stages.

The functional model iteration phase introduces an incremental and iterative approach. In every iteration, the content and the approach is planned. The results are analyzed for further iterations. Both analysis and coding are done: prototypes are built and the experiences gained are used in improving the analysis model. There are four main outputs in this phase: (1) prioritized functions – a prioritized list of functions that are delivered at the end of the iteration, (2) functional prototyping review document – collection of user's comments about the current increment, (3) non-functional requirements – listed as part of the scope for the next phase, and (4) risk analysis of further development – document that analysis encountered problems.

The design and build iteration is the main building phase. The output of the design and build phase is a tested system that fulfils at least the minimum agreed set of requirements. The design and build results are reviewed by the user in every iteration.

The implementation phase is where the finished system is shifted from the development environment to the actual production environment. In this phase the system is given to the users, which is also

being trained to use it. User manuals and project review reports are created and given to the user. DSDM defines four possible courses of further development. If the system achieves all requirements, no further work, or remodeling is needed. However, if a substantial amount of requirements has to be revised (for example, if they were not discovered until the system was in development), the process may be run through again from start to finish. If a less critical functionality has to be added, the process can be re-run from the functional model iteration phase.

2.2.2. Roles and responsibilities

The following roles as most significant in DSDM, as described by (Abrahamsson, Salo, Ronkainen, & Warsta, 2002):

Developers and senior developers cover all developer roles. Seniority is based on experience. The level of leadership in the team is based on the amount of seniority a developer has. The developer and senior developer roles cover all development tasks, such as analysis, design, programming and testing.

The technical coordinator role takes the task of defining the system architecture and is responsible for technical quality in the project. A technical coordinator is also responsible for technical project control.

The role of ambassador userrepresents duties that are to bring the knowledge of the user community into the project. This role is responsible forreporting the progress of the project to other users important for the customer and the project. By doing so, the ambassador user ensures that an adequate amount of user feedback is received. The ambassador user has to be a member of the user community that will use the system once it is completed. Since the ambassador user is unlikely to represent both the technical and the businessuser viewpoints, an additional role of advisor user is recommended. Adviser users are users who represent an important perspective from the point of view of the project. Adviser users are representatives of IT staff, or financial auditors.

A visionaryis a user participant with the most accurate perception of the business objectives of the system and the project. The Visionary is probably also the person with the initial idea to build the system and start the project. Two main tasks of the visionary are to ensure that essential requirements are found early on, and that the project keeps going in the right direction from the viewpoint of those requirements.

An executive sponsoris the person from the user organization who has the related financial authority and responsibility. The executive sponsor has the ultimate power in decision making.

2.2.3. Current research

DSDM was originally developed and continues to be maintained and researched by a consortium of several companies.(Tudor & Walter, 2006)describe how a large, process oriented, software organization coped the challenge of synchronizing agile with ISO 9001 certification through use of DSDM.Authors statedthat visual representation of a timeboxed plan helped team members know, at any time, what they were expected to complete. Also, developers pointed frequent meetings, small team size, and user involvement as helpful. The visual timebox sheets are one example of aaccepted practicein managing schedules. However, not all teams have been successful in shifting from traditional approaches to DSDM.

A case study of a partial DSDM adoption in a large organization is given by (Cobb, 2015). Another key feature of this project was the fact that the customer was the government, specifically, UK’s Ministry of Defense. The project’s objective was to create a sophisticated radar system which shows the position of the friendly forces in the cockpit of an aircraft. The key lesson learned on this project was to tailor the agile delivery technique in the project planning phase.

(Craddock, Richards, Tudor, Roberts, & Godwin, 2012) described a version of a DSDM which has been tailored specifically to complement Scrum. In this model, Craddock, *et al*. suggest using DSDM for project management and Scrum for product

development. The DSDM consortium also provides white papers, which describe DSMD usage with the Prince2 methodology.

(Bjelica, Mihić, & Toljaga-Nikolić, 2015)reviewed the success factors of IT projects and presented that in larger companies, only 9% of the projects are done on time and within the budget.Since DSDM's key principles are fixed time and cost, with changes allowed in the functionalities build, DSDM may prove helpful in finishing projects on time and cost. Furthermore, full implementation of the DSDM is more likely to be applied in big organizations, and on big projects. This is due to the fact that small companies cannot dedicate all the roles stated by the DSDM. However, partly implementing DSDM, for example, using the Moscow technique, iterative planning and including ambassador user and technical coordinator roles may help small companies finish on time and budget.

2.3. Scrum

The term "Scrum," representing a group of rugby players packing closely together trying to gain possession of the ball, first appeared in a study of two Japanese professors (Takeuchi & Nonaka, 1986). Takeuchi and Nonaka introduced a holistic approach, based on lean principles, which represented best practices in the Japanese industry. The Scrum model was then enhanced during the "Pasterur Project," which examined 50 highly effective software development organizations, at ATT Bell Labs (Sutherland & Schwaber, 2007).

The Scrum framework is very clearly defined in the Scrum guide by Ken Schwaber and Jeff Sutherland. The goal of Scrum is to deliver as much quality software as possible within a series of short time boxes called "sprints," which last about a month. Scrum is characterized by short, intensive, daily meetings of software development stakeholders. Scrum project planning uses lightweight techniques such as Burndown charts as opposed to Gantt charts and relies on self-organizing and cross-functional teams.

2.3.1. Process

Different authors describe various Scrum processes, which are very similar. The following review of a Scrum process is based on the 2016 Scrum guide. Scrum is conducted iteratively and products are developed incrementally, through time-boxed events named *sprints*, figure 4.Sprint's maximum duration is fixed to one month and its purpose is to provide a customer with a working piece of software which is releasable. A sprint consists of: (1) sprint planning meeting, (2) daily scrum, (3) sprint review, and (4) sprint retrospective.

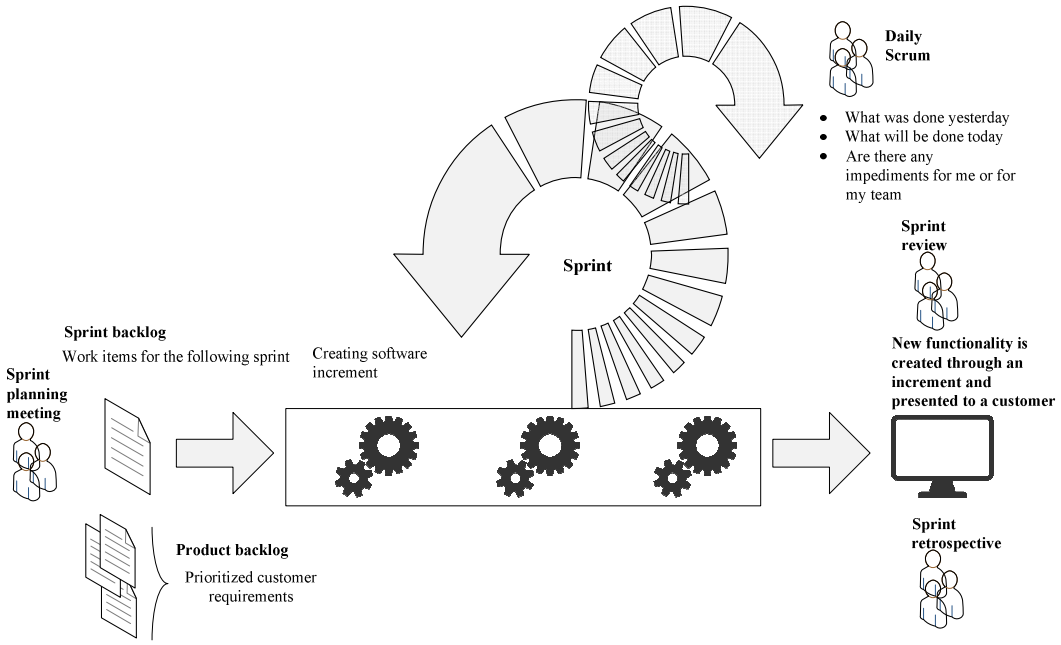

Figure 4: Scrum framework as presented in The Scrum Guide 2016

A sprint starts with a sprint planning meeting. In this meeting, the development team meets and creates a sprint backlog, from the items in the product backlog, which defines the scope of the work for the sprint. After the development team agrees on what needs to be done, they define the way they will do the required work.

The daily scrum is an everyday, fifteen minute, meeting in which the developers discuss what was done the day before, did they have any problems, and what will they work on next. The purpose of this meeting is to keep all the member informed on the work done, and work to be done in the sprint.

A sprint review meeting occurs at the end of each sprint. At this meeting, the entire scrum team meets together with the customer representative and demonstrates the functionalities they worked on during the sprint. Sprint review is the perfect time for the scrum team to gather the customer feedback and incorporate it into the subsequent sprint.

Finally, the sprint retrospective is the last meeting before the next sprint starts. This meeting is intended for the scrum team only and its goal is to make further enhancements in the scrum process. The team discusses various ways they can be more productive and effective. The Scrum master has a key role in this meeting, since he is accountable for the Scrum process.

2.3.2. Roles and responsibilities

Scrum roles and responsibilities have changed over time. The Scrum team is formed from three roles, as stated in the 2016 issue of the Scrum guide. These are:

Scrum master is responsible for ensuring Scrum is understood and enacted. Scrum masters do this by ensuring that the Scrum team adheres to Scrum theory, practices, and rules. The Scrum master also helps other, outside of the Scrum team interact with Scrum team. This coaching roles servers the project mainly by removing impediments to the development team's progress. Scrum masters can, but usually do not do any of the programming work.

The product owner is responsible for maximizing the value of the product and the work of the development team. One of the main responsibilities of a product owner is to manage the Product backlog (list of items which represent features, functions, requirements, enhancements, and fixes that make a product.) This means the product owner is accountable for ordering the items in the product backlog, making priorities, and ensuring the development team understands the items.

The development team consists of professionals who do the work of delivering a potentially releasable increment of the product at the end of each Sprint. One of the main features of a Scrum's development team is its cross functionality and self-organization. This means the development team has all the skills needed to develop the product and is empowered to choose the way they do so. The optimal development team size is three to nine members.

2.3.3. Current research

According to 10th State of Agile Report, Scrum, nearly 70% of respondents are using Scrum – Scrum (58%) and Scrum/XP hybrid (10%), making Scrum the most used agile approach.Regarding Scrum practices, (Mann & Maurer, 2005) found that daily meetings kept the customer up to date and that planning meetings helped in clarifying the development scope. Customers were more appealing to the development process as well. The customers stated that their satisfaction with the project that was based on Scrum was greater than with previous projects at the company. However, Mann and Maurer stress that the customer should be trained in the Scrum process so that they will understand the new expectations that the developers will have of them. It is also stated that the introduction of Scrum led to a reduction of overtime, and all developers recommended the use of Scrum in future projects. The study also showed that there is some time required for everyone involved to get used to the process. This resulted in longer sprints and unclear meeting agendas.

Also, Unlike XP, Scrum does not define any technical practices that can be call "best", however it does present a good project management framework. According to the (Akif & Majeed, 2012) there are some limitations to the Scrum framework, mention training, management, involvement, access to external resources, corporate or organizational size, sub contraction, developing large and complex systems as key research areas where no significant research has been done.

In the 2016 Scrum Guide, by Ken Schwaber and Jeff Sutherland, it is stated that: "Scrum's roles, artifacts, events, and rules are immutable and although implementing only parts of Scrum is possible, the result is not Scrum." It is against the definition of agile itself that you cannot change or adapt Scrum rules. This lack of flexibility in Scrum can become an obstacle to an organization shifting to Scrum. Adopting Ken's and Jeff's Scrum may lead to specific changes in the current organizational titles and roles. For example, the Project manager role is not defined in Scrum. Therefore, companies adopting agility for the first time should consider changing in small steps at first. For example, switching just one senior team to Scrum, and then pass on this experiences with the entire software development department.

2.4. Extreme Programming (XP)

Extreme programming originates from the Chrysler C3 Project in 1996. From then, XP was featured in a majority of agile studies (76%) as analyzed by (Dyba & Dingsoyr, 2008). XP has evolved from the problems caused by the long development cycles of traditional development models as evidenced by(Beck, 1999). XP model's main characteristics are short iterations with small releases and rapid feedback, close user participations, constant communication and coordination and collective code ownership. It consists of 10-12 practices, depending on the source, such as the planning game, pair programming, on-site customer, test-first programming, etc. The main body of knowledge for XP comes from (Beck, 1999).

2.4.1. Process

In the following figure, XP's phases are introduced according to (Beck, 1999). The life cycle of XP consists of six phases, figure 5: exploration, planning, iterations to release, productionizing, maintenance, and death.

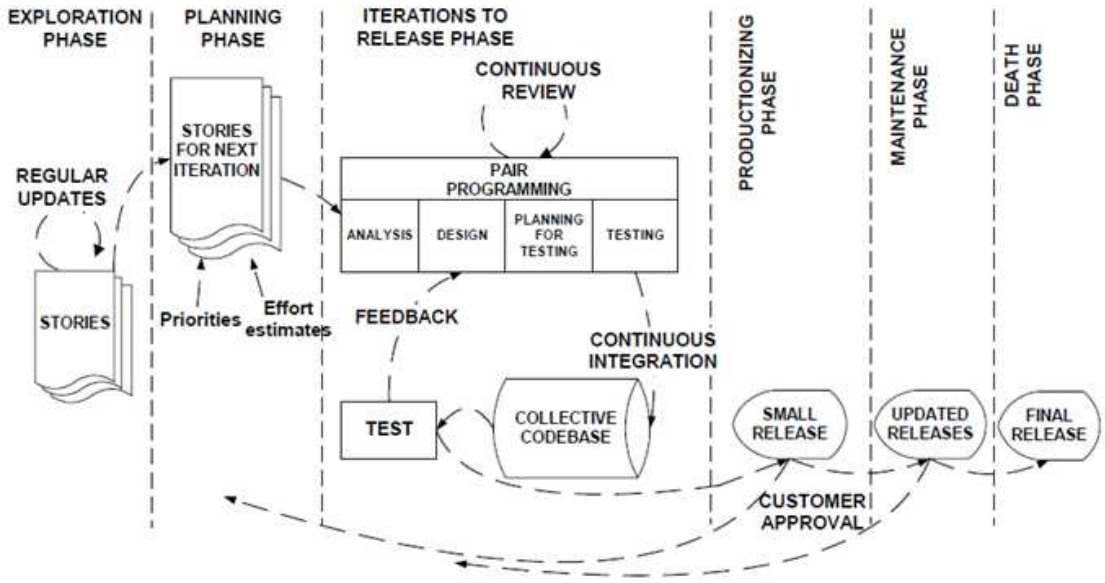

Figure 5: The life cycle of XP (Beck, 1999)

In the exploration phase, customers familiarize the development team with the product idea and write story cards to be included in the first releases. At the same time

the project team gets acquainted to the technology, tools and practices they will be using in the project. A prototype of the system is built. The exploration takes between a few weeks to a few months, depending on the familiarity of the technology used.

The planning phase sets the priority for the stories, the programmers estimate how much effort each story requires and determine the scheduled. Scheduling of the first release does not normally exceed two months. The planning phase lasts for a couple of days.

During the iterations to release phase, the schedule, set in the planning phase is broken down to a number of iterations that will each take one to four weeks to implement. During the first iteration, a system with the architecture of the whole system is created. At the end of the last iteration the system is ready for testing or production.

The productionizing phase consists of extra tests and checks of the performance of the system. At this phase, new changes may appear and the decision has to be made whether to includethem in the current release.The postponed ideas and suggestions are documented and can be implemented later during the maintenance phase, or some other project.

Throughout the maintenance phase, the XP team keeps the system in production and produce new iterations. This phase also requires specific customer support tasks. The maintenance phase may require incorporating new people to be introduced into the team.

The death phase is near when the customer does no longer have any new stories, requests or changes to be implemented. This requires that the system satisfies all of the customer's requests. This is the time when the necessary documentation of the system is written,since there are no more changes to the architecture, design or code. Death phase of a project may also occur if the system is not delivering the required or desired outcomes. In addition, the project will end if the system becomes too expensive for further development or maintenance.

2.4.2. Roles and responsibilities

The following roles and responsibilities are presented as described in (Abrahamsson, Salo, Ronkainen, & Warsta, 2002).

Programmers program the entire system, write tests and keep the program code as simple as possible. The first challenge inmanaging the team and making XP successful is communicating and coordinatingefficiently with other programmers and among team members.

The customer role writes the stories and functional tests, and decides when each requirement is satisfied. The customer is also responsible for setting the implementation priority for the requirements and giving feedback to the programmers.

Tester'sprimary activity is to write tests cases, together with the customer. Testers run functional tests, broadcast test results and maintain testing tools.

Tracker gives feedback about everyday work on a project. He traces time estimates made by the team and provides feedback on how accurate they are. The main idea behind this is to improve future estimations. Trackers inspect the progress of each iteration and evaluate whether the iteration goal is reachable. Trackers must do this with respect to the given resource and time constraints. They also alarm the team if any changes are needed to the process.

Coach is the person, who is well acquainted with XP and is responsible for the process as a whole. A sound understanding of XP practices, and experience in working with XP teams is important for this role. This enables the coach to guide the other team members in following the process.

Consultant is an external member to the team. His primary role is to advise the team with his technical knowledge. Consultant guides the XP team in solving their specific technical issues.

Manager role makes the decisions regarding the project.He communicates with the project team in order to understand the status of the project, its situation, and to distinguish any difficulties or deficiencies in the process.

2.4.3. Current research

Organizations shifting from waterfall to XP can quickly adopt its practices and achieve good results – 10% time reduction and 25% cost reduction,calculated by (Dyba & Dingsoyr, 2008). Dyba*et al.* also concludes that: (1) XP teams experienced improved communications, but were perceived by other teams as more isolated; (2)XP works best with experienced developers with domain and tool knowledge; and (3)XP leads to an increased collective code understanding and overall tacit knowledge improvements. A research, (Robinson H. , 2005), examining human, social and organizational factors related to agile development in three companies: a large multinational bank, medium-sized content security software company, and a small start-up company found that, despite the variations in physical settings and organizational structure, XP worked well in all three companies. Authors (Robinson & Sharp, 2001) identified good personality characteristics for members of XP development teams, stating trust as a key factor. Favorable team member traitis also: "analytical, with good interpersonal skills and a passion for extending his knowledge base (and passing this on to others)" as found by (Young, Edwards, Mcdonald, & Thompson, 2005).(Dyba & Dingsoyr, 2008)surveyed job satisfaction amongst employees in software companies that used XP and companies that did not use agile development methods. 95% of the employees who used XP answer that they would like their company to continue using their current development process, while the number for the employees in companies that did not use agile development was 40%.

As far as XP practices are concerned, numerous studies were conducted on this topic. A case study showed that participants had a divided opinion on pair programming, and that test first programming had good effects.A study done by (Ileva, Ivanov, & Stefanova, 2004) also showed mixed opinions on pair programming; It proved to be a very useful style of working with full respect to coding standards. However, working 40h a week in pairs required a lot of concentration which exhausted the developers. A studyby (Dyba & Dingsoyr, 2008) showed that 73% of the employees who used pair programming claimed that this practice speeds up the software development process. Also,the planning game was found to have a positive effect on collaboration within the company. The planning game process bridges the usual boundaries between project managers and software developers, as found by (Mackenzie & Monk, 2004).According to (Dyba & Dingsoyr, 2008), having a customer on site was suggested useful and had resulted in better collaboration with the customer.This practice benefited the customer side as well, since the customer had constant control over the process(Ileva, Ivanov, & Stefanova, 2004). XP practices result in stressful situations and long working hours in regards to the customer, as studied by (Martin , Biddle , & Noble, 2004). Martin *et al.* point that XP Customer practices achieve excellent results, but that they are also unsustainable, especially in long or high pressure projects. Martin *et al.* also studied the role of the customer in outsourced projects, and found that this was challenging because the customer was required to become acclimatized to the different cultures or organizations of the developers.

3. COMPARISON

Chronologically speaking, the Spiral model is the oldest method in the review, appearing in 1986, followed by Scrum and DSDM in the early nineties, and finally XP appeared in 1998. Based on the literature, all of the methods can be proclaimed as "active," since they are still used today, in both research and practice. Numerous systematic reviews of agile methods show that scrum and XP are leading as being the most used and cited methods, DSDM and Spiral have a far less presence in both organizations and studies. In the last decade, The Spiral model has been mostly used as a reference model when creating other models or choosing the appropriate one. Scrum and Spiral have one similarity, the featurethat they can be used as a process framework in which other practices can be used. This is also true for DSDM, however more case studies are needed for a stronger conclusion. For example, the developing team can use XP practices in their development. Table 1 summarizes the review of agile methods in section 2.

Table 1: Comparison of different agile methods

	Process	Roles and responsibilities	Current research
Spiral	Risk oriented process which can be transformed into other development processes depending on project characteristics.	Heavily dependent on experienced risk experts. Other than this, no roles are defined, hence it is hard to define a team required for the spiral process.	Currently Spiral is being researched as a process generating model. Not many applications of the Spiral can be found.
DSDM	An iterative and incremental approach that is more suited for large-scale projects, in which deadlines and costs are fixed and functions can be adjusted accordingly.	Defines roles on a project level, development team level and customer support level. All roles are well defined, but may not all be available in all environments, which can be a problem for practitioners.	DSDM's framework can enclose other methods, such as Scrum or Prince2.
Scrum	A lightweight iterative process which puts the focus on developing and project management, removing any obstacles the developing team may have, and having a potentially releasable increment, approved by the customer, in every iteration.	A self-organized and cross-functional team which has the power to do the work the way they find appropriate. The Scum process is ideal for organizations with small number of hierarchical levels.	Mostly used method in software development today. Along with XP it is also one of the most researched method.
Extreme programming	A developer oriented process which places the needs of software developers in the first place.	XP defines all roles required in a development team. It can be a good reference point for startups and newly created development teams.	At the moment, there are mixed opinions on the effects of the XP. It is suggested that it should be used with experienced teams.

DSDM introduces several user roles, business advisor, which represent different customer viewpoints, whereas XP defines only one role for user and Scrum recommends that "key stakeholders" be present at sprint reviews. It can be concluded that these three methods recognized the need for a user role to be involved in the development process. The Spiral model has no mention of a user or customer role in its original work. Regarding team size, XP and Scrum recommend using small teams, preferably less than 10 developers. DSDM and Spiral on the other hand, have examples of being used in large projects, however teams were again not bigger than 20 developers. Spiral and XP empower their teams to make decisions on their own. Thus, adopting one of these methods requires a cultural change in the organization. (Nerur, Mahapatra, & Mangalaraj, 2005) discuss the organizational factors affected in adopting agile development.. The 10th State of Agile Report also shows 10 leading causes of failed agile projects and barriers to further agile adoption. Not all agile methods are suitable for all phases of the software development life cycle, as elaborated by (Abrahamsson, Salo, Ronkainen, & Warsta, 2002; Abrahamsson, Warsta, Siponen, & Ronkainen, 2003). Figure 6based on the model described by Abrahamsson *et al.* shows which phases of software development are supported by different agile methods:"*Each method is divided into three blocks. The first block indicates whether a method is suitable for project management. The second block identifies whether a process, which the method suggests to be used, is described within the method. The third block indicates whether the method describes any concrete guidance, activities and work products that could be followed and used under varying circumstances. A grey color, in a block, indicates that the method provides and white color indicates that the method does not provides detailed information about one of the three areas evaluated*".

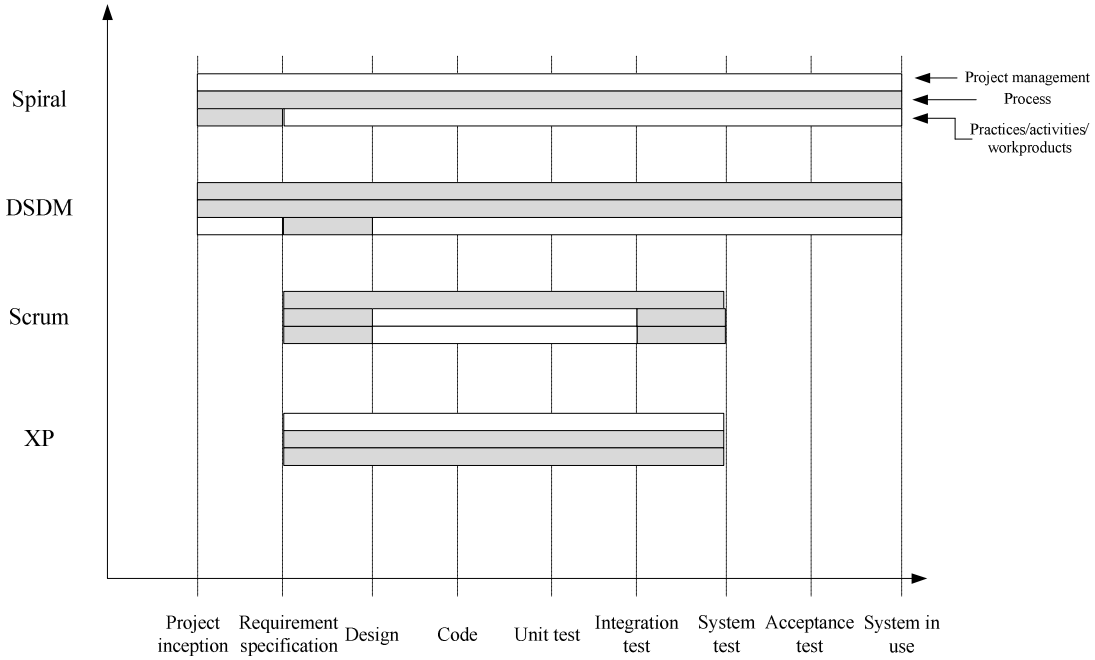

Figure 6: Comparing life cycle, project management and concrete guidance (Abrahamsson, Salo, Ronkainen, & Warsta, 2002; Abrahamsson, Warsta, Siponen, & Ronkainen, 2003)

Concerning project management, XP does not provide any guidance. However, (Abrahamsson, Warsta, Siponen, & Ronkainen, 2003)provide examples of XP project management literature. Unlike XP, Scrum covers project management and it is focused on managing agile software development projects. Since Scrum does not give explicit practical examples of how to develop software, the same authors give an example of how Scrum is combined with XP. DSDM provides a framework which supports rapid application development and development teams work facilitation in an ever-changing environment. The Spiral model does not have any project management practices described. One of the reasons for this is due to fact that the Spiral model has the smallest body of knowledge.

The DSDM provides a full coverage over the development life-cycle. Neither XP nor Scrum cover project inception phase or provide any detail on how to accept the product and deal with it once it is in use. Scrum also does not give any information regarding the development phase. The Scrum guide does not explicitly mention integration or maintenance phase, but it does state that a software project lasts until the product backlog – "As long as a product exists, its Product Backlog also exists." This is why (Abrahamsson, Salo, Ronkainen, & Warsta, 2002) suggest somemethods require a complementing approach to support software development.

Concrete guidance, in the context of figure 6, refers to practices or activities that provide a specific guidance on how a specific task can be executed.XP's main focus is on software development practices. Its purpose and goal is to share best software development practices". These practices are described in detail in (Beck, 1999). However, it is out of the scope of this paper whether these practices have any value. (Abrahamsson, Warsta, Siponen, & Ronkainen, 2003), and (Recker, Holten, Hummel, & Rosenkranz, 2017) provide a more detailed analysis on this question.The DSDM method states that due to the fact that each organization is different it cannot define practices that would fit all of them as described in (Abrahamsson, Warsta, Siponen, & Ronkainen, 2003). Instead organization should develop practices themselves. DSDM does not provide any guidance on this, however it does give examples of DSDM used

in various organizations through white papers available on the community website. One technique that DSDM proposes is the Moscow technique for requirements prioritization (DSDM Consortium, 2008; Cobb, 2015). (Abrahamsson, Warsta, Siponen, & Ronkainen, 2003) state that Scrum as well provides concrete practices for the requirements specification phase and integration testing phase. The Spiral model provides a template table which can be used in the project initiation phase as well as when entering any later phase. This table is described in more detail in (Boehm B. W., 1986).

(Abrahamsson, Warsta, Siponen, & Ronkainen, 2003) conclude that the agile community, literature, and developers are being driven more by abstract principles than by concrete guidance and practices – "The agile community is more concerned about getting acceptance to proposed values than in offering guidance on how to use the operative versions of these values." The more detailed and practice oriented methods, such as XP, are very limited in their scope. More work is needed in determining how the described methods can be used in different organizations and situations, so that practitioners have a solid knowledge base on which to make decisions.

In summary, all of the reviewed agile methods are placing an emphasis on the following aspects: (1) delivering something useful, (2) reliance on people, (3) encouraging collaboration, (4) promoting technical excellence, (5) being constantly adaptable (Abrahamsson, Salo, Ronkainen, & Warsta, 2002). In this paper another aspect is mentioned – doing the simplest thing possible, more appropriate meaning would be doing the most important thing possible, since all of the methods perform prioritizations.

4. CONCLUSION

It can be noted that all software development models described in this paper, were created, developed, and tested in a particular organization. For example, the Spiral model was implemented in TRW, the Scrum in Bell Labs, Extreme Programming in Chrysler. The important word here is developed and tested: it is through trial and error, and with respect to the organizational environment, technology, resources, and to specific project/s, at that time, that these models incurred. Therefore, it would be a common mistake to simply copy an already given process methodology and apply it to a particular organization. This may work, but the chances of it being effective and productive are slim. It is much more productive to study these models and use the principles and values they promote as inspiration for a custom organizational software development method. This goes in accordance with all the models and methodologies created by big international IT companies (Microsoft's Sure Step, Oracle's OUM, IBM's Fastlane, etc.) and with the 10th State of Agile Report which states that 30% of the surveyed companies use a mixture of agile models.It is important that people involved in defining the organizational PM procedure get acquainted with these process methodologies.One of the best lessons project managers, or team leaders on software projects can learn from the agile movement, is that they have to find a way to use whatever resources as well as processes, which are available at the moment. This knowledge is also vital since in the last two decades, the debate about agile methods has been very significant and researchers and practitioners are not aware of existing approaches or their suitability for varying real-life software development situations as evidenced by (Abrahamsson, Warsta, Siponen, & Ronkainen, 2003).

Future work should be in the field of hybrid models which combine agile and traditional development practices with project management approaches. (Abrahamsson, Salo, Ronkainen, & Warsta, 2002) mention different authors which share the opinion that no single development approach can conform to the whole spectrum of different projects. Boehm, the author of the hybrid Spiral model, recognizes in his work (Boehm B, 2002) that: "each approach has a home ground of project characteristics within which it performs very well, and much better than the other, outside each approach's home ground, a combined approach is feasible and preferable."

Future work should also be in the field of creating methods and models, which will help organizations analyze the characteristics of

the project and the project environment and as a result propose a method for the particular project. Examples of this new research can be found in Boehm's new works regarding the Spiral model, (Boehm &Belz, 1989; Boehm & Turner, 2015).

REFERENCES

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). *Agile Software Development Methods: Review and Analysis.* Oulu: VTT Elektroniikka.

Abrahamsson, P., Warsta, J., Siponen, M. T., & Ronkainen, J. (2003). New Directions on Agile Methods: A Comparative Analysis. *Proceedings of the 25th International Conference on Software Engineering* (str. 244-254). Portland: IEEE Computer Society Washington.

Akif, R., & Majeed, H. (2012). Issues and Challenges in Scrum Implementation. *International Journal of Scientific & Engineering Research, Volume 3, Issue 8*, 1359-1362 .

Beck, K. (1999). *Extreme Programming Explained: Embrace Change* . Boston: Addison-Wesley Longman Publishing Co.

Bjelica, D., Mihić, M., & Toljaga-Nikolić, D. (2015). Theoretical perspective of IT project management approaches, success factors and maturity models. *Serbian Project Management Journal*, 43-55.

Boehm, B. (2002). Get Ready for Agile Methods, with Care. *IEEE Computer Journal*, 64-69 .

Boehm, B. W. (1986). A Spiral Model of Software Development and Enhancement. *ACM SIGSOFT Software Engineering Notes*, 14-24.

Boehm, B., & Belz, F. (1989). Experiences with the spiral model as a process model generator. *Proceedings of the 5th international software process workshop on Experience with software process models* (str. 43-45). Kennebunkport: IEEE Computer Society Press.

Boehm, B., & Turner, R. (2015). The incremental commitment spiral model (ICSM): principles and practices for successful systems and software. *Proceedings of the 2015 International Conference on Software and System Process* (str. 175-176). New York: ACM.

Cobb, C. G. (2015). *The project managers guide to mastering agile: Principles and Practices for an Adaptive Approach.* Hoboken, New Jersey : John Wiley & Sons.

Craddock, A., Richards, K., Tudor, D., Roberts, B., & Godwin, J. (2012). *white papers.* Preuzeto sa Agile business: https://www.agilebusiness.org

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *The Journal of Systems and Software*, 1213-1221.

Dragan, B., Marija, T., & Zorica, M. (2013). Risk Appraisal for software projects in accordance with project management maturity models. *Serbian Project Management Journal*, 35-43.

DSDM Consortium. (2008). *DSDM Atern Handbook.* Kent: DSDM Consortium.

DSDM Consortium. (2014). *The DSDM Agile Project Framework Handbook.* Kent: DSDM Consortium.

Dyba, T., & Dingsoyr, T. (2008). *Empirical studies of agile software development: A systematic review.* Trondheim: Inform. Softw. Technol.

Hendrix, D. T., & Schneider, M. P. (2002). NASA's TReK project: a case study in using the spiral model of software development. *Communications of the ACM - Supporting community and building social capital*, 152-159.

Highsmith, J., & Cockburn, A. (2001). Agile Software Development: The Business of Innovation. *IEEE Computer Journal*, 120-122 .

Ileva, S., Ivanov, P., & Stefanova, E. (2004). Analysis of an agile methodology implementation . *Proceedings of 30th Euromicro Conference* (str. 326-333). IEEE Computer Society Press.

Jennifer, S., & Peter, C. (1997). *DSDM: Dynamic Systems Development Method: The Method in Practice.* Boston: Addison-Wesley Professional.

Larman, C., & Basili, V. R. (2003). Iterative and Incremental Development: A Brief History. *IEEE Computer Journal*, 47-56.

Mackenzie, A., & Monk, S. (2004). *From Cards to Code: How Extreme Programming Re-Embodies*

Programming as a Collective Practice. Computer Supported Cooperative Work, Volume 13, Issue 1.

Mann, C., & Maurer, F. (2005). A Case Study on the Impact of Scrum. *Proceedings of the Agile Development Conference* (str. 70-79). Washington: IEEE Computer Society.

Martin , A., Biddle , R., & Noble, J. (2004). The XP customer role in practice: three studies. *Proceedings of the Agile Development Conference* (str. 42-54). Washington: IEEE Computer Society.

Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of Migrating to Agile Methodologies. *Communications of the ACM - Adaptive complex enterprises* , 72-78.

Recker, J., Holten, R., Hummel, M., & Rosenkranz, C. (2017). How Agile Practices Impact Customer Responsiveness and Development Success: A Field Study. *Project Management Journal*, 99-121.

Robinson, H. (2005). Organisational culture and XP: three case studies. Proceedings of the Agile Conference (ADC'05).

Robinson, H., & Sharp, H. (2001). *The characteristics of XP teams, in: Extreme Programming and Agile Processes in Software Engineering.* Berlin: Lecture Notes in Computer Science, vol. 3092, Springer.

Sutherland, J., & Schwaber, K. (2007). *The Scrum Papers: Nuts, Bolts, and Origins of an Agile Process.* Cambridge: Scruminc.

Takeuchi, H., & Nonaka, I. (1986). The New New Product Development Game. *Harvard Business Review* , 137-146.

Tudor, D., & Walter, G. A. (2006). Using an Agile Approach in a Large, Traditional Organization. *Proceedings of AGILE 2006 Conference (AGILE'06)* (str. 367 - 373). Washington: IEEE Computer Society.

Young, S., Edwards, H., Mcdonald, S., & Thompson, J. (2005). Personality characteristics in an XP team: A reperitory grid study. *Proceedings of Human and Social Factors of Software Engineering.* St. Louis.