ISSN 1392–0758 SOCIAL SCIENCES. 2014. Nr. 3 (85)

Hybrid Approach in Project Management – Mixing Capability Maturity Model Integration with Agile Practices

Artur Ziółkowski

Gdańsk University of Technology
Narutowicza 11/12, 80-233 Gdansk, Poland

Tomasz Dęręgowski

Acxiom Corporation
Wołoska 3, 02-675 Warszawa, Poland

crossref http://dx.doi.org/10.5755/j01.ss.85.3.8416

Abstract

This paper introduces an idea of hybrid approach in managing software development projects. The main goal of this research is to prove that it is possible to design a consistent method for managing software development projects which is based on different corporate standards and methods. The authors also want to show that this new hybrid approach is beneficial for IT organization, triggers synergy effects and brings software development process to a higher level, impossible to achieve when methodologies and standards are used separately. This paper discusses an exemplary implementation of hybrid management process which is based on CMMI and Scrum.

Keywords: **project management, standards, methodologies, CMMI, Agile, Scrum.**

Introduction

The software development process is implemented by all types of organizations, both software houses and organization which develop software to support their main business areas (e.g. banks). Over the past few years companies developed many different methodologies for managing software development processes. Some of them became corporate standards and are widely used in IT industry.

Based on the level of complexity, software development methodologies could be classified into two main categories: heavy (traditional) and light (agile) (Philips, 2007; McMahon, 2010). Heavy methodologies are prescriptive, with numerous rules to follow, many roles defined and artefact-intensive. They assume that a formal, detailed process which precisely defines all aspects of software development project is the key factor guaranteeing the success of a project (Parth, 2007). The success of a project is understood as delivering a product within the scope defined at the beginning of the project, within the budget and on schedule. Good examples of heavy methodologies are PMI, PRINCE2 and RUP (see Bergstom, 2003; Kruchten, 2004; PMI, 2008; PRINCE2, 2009).

Light methodologies are adaptive with fewer rules to follow. They are based on synergy and self-organizing teams. They assume that adapting changes in client's requirements (Wiegers, 2003) during the whole project is a guaranty of project success. It is not important to deliver a project on schedule, within budget and scope. A project is successful when at the end of it stakeholders receive the product they need. The most popular of light methodologies are Scrum, Kanban, XP and Lean.

In addition to using software development methodologies, many companies introduce process improvement programmes in order to optimize organization underlying processes and become more efficient (Bass, 2003). These programs define which process and why should be applied to successful organization, but they do not define how they should be implemented. Examples of such standards are CMMI, ITIL and TOGAF.

The choice of method for managing IT processes which corresponds to organization's needs and is adequate to its unique culture is a strategic and one of the most important decisions to be made by an IT organization. It impacts the way organization works and would be one of the organization's key success factors. When choosing a method of managing IT organization, several factors need to be taken into consideration:

- Human factor represented by the client and the team providing IT solution for the client.
- Technological factor represented by all technological aspects related to product development.
- Communicational aspect related to transforming client expectations and needs into product functionalities.

In addition to the above factors, which have direct impact on the project, there are also factors that shape the project environment in an indirect way. Management of an organization, external suppliers, legal conditions, etc. –

they all have an impact on project's course and should be taken into consideration.

All these factors shape a unique, multifaceted environment dependant on many factors with non-predictable behaviour. Methodologies and standard processes are designed to bring order into this complex and shaky environment and minimize all types of risks.

Because of its complexity and multidimensionality, every IT project is unique. The authors' experience shows that it is impossible to develop a universal method for managing software projects which would be suitable for different projects with diverse specificity. Management of software development processes is part of situational management (Griffin, 2012) (situational approach to management). A more suitable approach is to define a unique set of rules and procedures for each newly started project. These rules should be adapted from all type of methodologies, both heavy and light.

The solution suggested by authors is in contradiction to the standards of the IT industry. Most companies introduce single methodology in whole company and use it to manage all types of projects. What is more, many organizations treat traditional and Agile methodologies as mutually exclusive and assume that it is impossible to use both of them in the same organization (Schwaber, 2002). Such an approach is counterproductive and decreases origination effectiveness.

1. The uniqueness of projects and management standards

There are many examples of projects which used traditional process approach (e.g., CMMI) and were successful. There are also many examples of projects which were successful in using Agile techniques. According to situational management principles, the specificity of particular projects (Schwalbe, 2010), their entropy and complexity determine which approach is more suitable.

The authors postulate the elaboration of a new, adaptive and situational model of IT organization which assumes a dynamic creation of unique management process for each newly established project. This new approach should be based on a variety of IT standards, both traditional and Agile. It would enable achieving a synergy effect.

Decisions on how to implement each part of software development process for a particular project should be based on project's specificity. A unique set of project parameters like type of the client, size of development team, experience in particular technology and, in particular, type of the product should be taken into consideration.

2. Existing methods for managing the process of software development

As mentioned above, two approaches have dominated the methods of software development management: heavy (traditional) and light (agile). In further research the authors concentrate on two methodologies: CMMI (Capability Maturity Model Integration), which is often treated as a representative of heavy approach and Scrum, a sample Agile approach. The authors have decided to concentrate on these two methodologies because of their experience in introducing Scrum practices in hierarchical and structural CMMI organization.

2.1. Capability Maturity Model Integration as an example of traditional approach in software development management

CMMI is a process improvement method which allows integrating all process and procedures existing in an organization and identifying potential gaps. The CMMI model was founded by the United States Department of Defence (DOD) and is implemented mainly in companies which work for army, governmental agencies or large corporations. In such types of organizations the most important factors are confidentiality, security and stability, rather than the price or ability to adapt to client's changing needs. CMMI is usually implemented in a high-cost of failure domain.

The CMMI model is often used to assess the maturity of the process implemented in the organization. CMMI model defines five levels of maturity (Chrissis, 2011):

- Initial (Level 1) – on initial level processes are chaotic and undocumented. Success in such an environment is still possible, but mainly because of team members' knowledge and their dedication. What is more, the success will be hard to repeat.
- Managed (Level 2) – key processes are planned and run according to established policies (they are monitored, controlled, they results are reviewed, adherence to process description is evaluated). Depending on the project, processes definitions and descriptions differ. CMMI defines seven Process Areas that need to be implemented in an organization so it may be considered as Managed in CMMI terms.
- Defined (Level 3) – all projects in an organization use the same set of standards. In comparison to Managed Maturity level, scope of defined processes is wider and their description is more detailed. To consider organization as Defined it needs to be Managed and have implemented additional eleven Process Areas.
- Quantitatively Managed (Level 4) – processes are measured and controlled. The data and metrics related to processes performance are collected and analysed. Thanks to historical estimates, quality and performance metrics, it is possible to predict process performance. To consider organization as Quantitatively Managed it needs to be Managed, Defined and have implemented two additional Process Areas.
- Optimizing (Level 5) – Organization is continuously improving its performance through technical innovations, defect prevention and processes review and assessments. To consider organization as Optimizing it needs to be Managed, Defined and Quantitatively

Managed and have implemented additional two Process Areas.

CMMI defines which processes should be implemented and why but it does not define how they should be implemented. Thus iterative approach and waterfall life cycle are equally consistent with CMMI.

Identifying CMMI with waterfall approach is due to its origins (CMMI was founded by DOD) and to the way it is usually implemented. Compliance to CMMI is often one of the requirements in contracts held by army, government agencies or multinational corporations. Usually this is the only reason why companies implement CMMI – they want to tender for a contract. They are not interested in raising the quality of their processes. In such cases organization introduces standard, generic and usually waterfall processes which are not adjusted to processes and procedures already existing in an organization. Such processes are not optimal; sometimes they can even worsen organization's performance. They are implemented in a standard manner because they increase the probability of passing SCAMPI appraisal.

2.2. Scrum – an Agile Project Management approach to Software Development

Scrum is one of the most popular and well known Agile methodologies. Many Scrum elements such as iterations, incremental software development, self-managing teams and adaptation to changing requirements are common to other Agile methodologies (Sutherland, 2010; Schwaber, 2011).

Iterative approach means that the duration of the project is divided into parts called iterations or Sprints. Every Sprint is organized in the same way: at the beginning of the Sprint the team plans which features will be developed during the Sprint; then team members develop these features and at the end of the Sprint the team reviews the features created during the Sprint with clients and get their feedback. Sprint usually lasts from two to four weeks.

A product is created incrementally, each Sprint we deliver a complete set of functionalities accepted by the client. Scrum is adaptive because it lets the team react to constantly changing requirements, market situation, changes in project team and others. A modification of project plan and project scope does not require contract renegotiation; changes are adapted on an ongoing basis.

The described features of Scrum have one common goal: to make cooperation with client smoother. Agile methodologies treat the customer as a partner, as member of the team. The customer actively participates in development process and can impact the course of the project throughout its duration.

2.3. Differences between CMMI and Scrum

Scrum and CMMI represents two different approaches to Software development. CMMI is prescriptive, defines many rules to follow, processes are well defined and should be implemented by the book. Scrum and other Agile methodologies are adaptive; they are sceptical on process definition and are based on common sense rather than on strict, detailed processes definitions. Agile methods value people and interactions between them over processes and tools.

Because of the importance of processes, CMMI organizations are usually heavy documented, whereas in Scrum we distinguish several types of documents which usually contain graphical representation of information rather than plain text.

In CMMI organization management and formal corporate structure plays important role, whereas Scrum teams prefer flat structure, self-organizing teams and authority based rather on skills and experience than on formal, corporate titles. Traditional managers in Scrum organizations try to coach the teams; they do not manage them in a traditional way. Their main focus is on removing impediments and eliminating barriers to progress.

Also the way in which CMMI and Scrum are implemented differs. Usually introducing CMMI is a top-down initiative launched by higher management, whereas introducing Scrum is usually a grass root effort inspired by engineers and techies.

These differences often lead to an assumption that CMMI and Scrum are self-excluding approaches. According to authors' knowledge and experience, this is not true. Both CMMI and Scrum have consistent goals: higher quality, happier customer, shorter time to market. Both approaches can be extremely effective and benefit project and organization performance.

3. Overcoming CMMI and Scrum shortcomings

CMMI DEV was implemented on highest maturity level by many successful companies such as Samsung, the Boeing Company, HP Enterprise Services, IBM Global Business Services, Lockheed Martin, and others (CMMI Institute, 2013). In Poland CMMI DEV v1.3 was introduced in companies such as Asseco Poland SA, Atos Belux, Alcatel-Lucent, HSBC Global Technology Centre Poland and HIS Global (CMMI Institute, 2013).

Scrum was implemented by major IT companies such as Google, Yahoo, Microsoft, Facebook, Adobe, Nokia, Siemens, BBC, CNN, General Electric, Bank of America, and Novell (Google, 2013).

They are many examples of successful projects which were managed with CMMI or Scrum. This does not change the fact that there are many well know shortcomings of both methodologies. What is crucial in the context of this article, many CMMI weaknesses can be overcome by using some Scrum practices. Also, Scrum can benefit from using some of the practices described in the CMMI model. Complementarity of Scrum and CMMI can be achieved through their parallel implementation in the same organization.

Not only software houses, but also their clients, can benefit from this new approach. On the one hand, clients expect that vendor organization will have implemented CMMI. From the client's perspective it should guarantee high quality of development process which will result in high quality of the product. On the other hand, customers want to participate in the development process on daily basis. They request direct impact on developed product and ability to introduce new requirements and change existing ones at every stage of the process.

Mixing CMMI and Agile into one process may help satisfy both client's needs and create process which is mature and dynamic at the same time. Due to this, we can achieve a synergy effect and build a flexible and Agile process on a solid CMMI foundation, create a mix of models and methods, with selected techniques adopted from CMMI and Scrum, to troubleshoot specific challenges.

3.1. Overcoming CMMI drawbacks with Scrum

CMMI is defined on high level of abstraction. Sixteen out of twenty-two CMMI for Development Process Areas are common to other CMMI models (CMM for Services and CMMI for Acquisition). They are defined without many details related to software development process because they may be implemented for different purposes in varied environments. Definitions of many CMMI Process Areas contain statements such as 'use proper technique' or 'use proper tool' without naming them. CMMI authors intentionally pass the decision to choose proper tools and techniques to the team which is introducing CMMI. It is also important that CMMI authors admit that many good practices have been omitted because they did not fit the general, high-level, CMMI concept.

As noted in Chapter 2.1, CMMI is process-agnostic: it defines what should be done and why but it does not say how. The way how to implement a particular Process Area (or its parts) can be taken from any methodology. In this article the authors will concentrate on filling CMMI gaps with Scrum tools and techniques.

A good example of such filling may be CMMI requirement to control the progress of work by Project Manager. This requirement is defined in Project Monitoring and Control (PMC) Process Area. The main objective of PMC Process Area is to monitor project progress and check its compliance with the schedule. Anomalies and inconsistencies with the plan should be detected and proper corrective actions should be taken.

Nowadays projects depend upon engineers with specialized, technical skills. Project managers, even with technical background, will never have sufficient knowledge and experience compared to the expertise of the members of the teams they lead. In consequence, they are not able to effectively track the progress of technical tasks. This problem was pointed out by Peter Drucker (1957), the leading management thinker of the twentieth century. He believed that it is not possible to directly control the work of knowledge workers – engineers with specialized, technical skills. Knowledge workers cannot be managed in traditional manner, they must manage themselves.

CMMI does not define appropriate mechanisms and tools that allow knowledge workers manage themselves while reporting progress to higher management. Such tools could be found among Agile practices. Scrum defines the concept of self-organizing teams where team members assign tasks themselves, control the progress of work, and report statuses on daily basis. It also gives appropriate tools which allows putting the concept into practice. Sprint Planning Meetings and Daily Scrum Meetings let practitioners plan work, assign tasks and track the progress of work on a daily basis.

Project Monitoring and Control is not the only Process Area described by CMMI which lacks implementation details. Thus, to implement CMMI Quantitative Project Management Process Area, we can use one of Agile metrics – velocity – which measures the rate at which the team performs work and how this rate is changing over time.

Another example may be CMMI Process Area Organizational Performance Management and Scrum Retrospective Meeting. A Scrum Retrospective Meeting may be used to gather information regarding problems a team is struggling with and about potential improvements which may be introduced. These types of activities are required by CMMI Organizational Performance Management Process Area.

The above examples are part of many examples which show that goals set by CMMI could be achieved with methods and tools adopted from Scrum. One of the most spectacular examples of introducing Scrum in CMMI organization is implementation made my Jeff Sutherland in Dutch company Systematic Software Engineering. Mixing Scrum and CMMI within a single organization gave spectacular results:

- Productivity in large teams was increased by 100 %,
- The cost of the project was decreased by 50 %,
- Average estimates inaccuracy level was 10 %,
- The rate of project completion within budget and schedule was 92 % (Sutherland, 2010).

After introducing Scrum, the status of CMMI Level 5 organization was preserved.

Scrum can give CMMI teams much more than just a way to implement some of CMMI practices. Scrum can make CMMI more innovative and keep it up to date with new trends in IT industry. The process of developing CMMI is formal and it takes years to release new version of CMMI model. CMMI is a successor of CMM model which was developed from 1987 until 1997. The first CMMI model Version 1.1 was released in 2002. CMMI Version 1.2 was released in 2006 and CMMI Version 1.3 in 2010. CMMI is evolving, but it does not evolve quickly enough to keep up with a rapidly changing IT industry. This problem may be overcome by introducing some of Agile practices to support CMMI goals. Agile community is much more active and innovative and adopts new trends very quickly. Adding some of Agile practices to CMMI environment can help it adapt most recent trends. A good example of such use of Agile tools in CMMI environment can be introducing Extreme Programming (XP) practices. By definition CMMI for Development is designated for IT teams. Unfortunately, CMMI lacks engineering practices that would allow development teams produce code faster and more efficient, with higher quality and less number of defects. Such practices are a part of XP, software development methodology which is intended to increase the quality of software development process and software products (Beck, 2004). Example of XP practices are Pair Programming, Test Driven Development, Continuous Integration, Small and Constant Releases, Coding Standards, Collective Code Ownership and more. Introducing some of these practices could significantly increase efficiency and quality of developed code which are CMMI goals.

Being innovative also means experiments. Development teams should test different approaches, processes and metrics and choose only those which correspond to the need of a particular team and particular project. Agile mindset encourages experimentation which cannot be said about the CMMI model. CMMI teams often do not want to try new things because they are afraid of losing SCAMPI appraisal. Agile can help and encourage CMMI to iteratively introduce improvements. Introducing improvements one by one, not all at the same time has many advantages. The most important is the opportunity to measure how the process has changed after introducing a particular practice. If we introduce several practices at the same time, it is hard to define which of them has changed the process performance. When a single practice is introduced, we can easily measure its impact on the process. Sequential introduction of improvements has another important advantage. It reduces resistance and fear of change. Innovations are isolated, we change single process element, not the process as a whole. We can also use iterative approach when introducing CMMI in a new company. Introducing CMMI does not have to indicate revolution. Due to the fact that Process Areas are introduced iteratively, the team has a chance to get familiar with particular Process Area before introducing others.

The CMMI organization often has problems with engaging clients in daily project work. Such engagement is especially important in today's globalized world where more and more contracts are executed by companies from different countries, working on different continents. Building trust in such an environment is not easy and traditional CMMI approach where client is not a direct contributor to the evolution of product is not sufficient. Clients do not want to rely on contract relationship; they want to be a part of development process and shape the product on daily basis. Such relationship can be easily implemented with some of Scrum practices like Sprint, Sprint Planning Meeting and Sprint Review Meeting. In Scrum clients are active members of the team. They have a direct impact on priorities and requirements, and their feedback is collected and adapted on daily basis.

The last but not the least benefit that CMMI teams can get from agile approach is simplicity. Quite often CMMI documentation is overwhelming. The average CMMI Level 3 SCAMPI Appraisal examines over 400 document types and over 1000 artefacts (Dalton, 2011). Agile is not questioning the sense of writing documentation but it requests writing documents only when they are read by someone. The average Scrum project produces 39 artefacts (Dalton, 2011). Using Agile experience can encourage CMMI teams to reduce the number of held documents and treat as documentation other, not obvious artefacts like code comments, digital photos of Scrum board, whiteboard drawings, etc.

Using agile approach in documentation also means being minimalistic, pragmatic, creating optimal documents and documents templates and, what is more important, improving them all the time. Policy documents may be brief, not longer than one page, and still contain all important information. Same about procedures, meeting agendas, and meeting notes – they should be short and contain as little detail as possible. Before adding all the details by default organization should rather wait for people to ask for such details. Keeping documentation short increases the probability that it will be used by people.

As seen in the above examples, Scrum and other agile practices can support CMMI on many different levels. What is crucial, Scrum values are not in contradiction with the CMMI model. On the contrary, Scrum supports CMMI goals, can help implement some Process Areas and make CMMI more effective and efficient tool.

3.2. Overcoming Scrum drawbacks with Scrum

In many organizations which have implemented Scrum, the software development process is reactive. Such organizations react to most recent client needs but they quite often miss a longer-term view. Figure 1 shows a graphical representation of levels of planning in an organization which develops software.

We distinguish six levels of planning, some of them are supported by Scrum, some not:

- Day – plan Team work for a single day. This task is performed by Scrum Master and Team. To track daily progress, Scrum introduces Daily Stand-up Meeting.

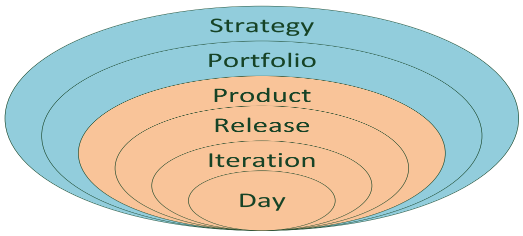

Figure 1. Level of planning in the organization which creates software

- Iteration – plan Team work for single iteration (time period from 2 to 4 weeks). This level of planning is handled by Scrum Planning Meeting, Scrum Review Meeting and Scrum Retrospective Meeting.
  Product Owner decides which tasks should be performed during the Sprint, Team decides how much work it is able to handle during the Sprint.
- Release – release covers several iterations. Product Owner decides which features should be included in particular release and prioritizes them appropriately in Product Backlog.
- Product – all estimated requirements (functional and non-functional) for the single software product are collected in Product Backlog. Product Owner via priorities directs project progress. She chooses which features and in which order should be implemented. She also decides about the scope of each feature.
- Portfolio – on the portfolio level, the organization manages the overall product offering (products and services) and dependencies between them. Scrum by definition does not offer tools to manage this level of planning.
- Strategy – on the strategy level, the company defines what it wants to be, its strategic goals and visions, the direction it wants to follow during the next several years. Scrum by definition does not offer tools to manage this level of planning.

As shown above, Scrum supports four levels of planning; two levels (Portfolio and Strategy) are not backed by Scrum. This gap can be neutralized by some CMMI practices, especially those defined on the third, fourth and fifth Maturity Level. On these levels CMMI looks beyond the needs of a single project. It standardizes processes and tools so they can be used in different, not related projects, it helps to measure and improve the performance of organization as a whole and lets it become less wasteful and leaner. Such strategic initiatives might not be profitable from the perspective of single projects. But when we look at them from the perspective of whole organization, all its current and future initiatives, such actions may prove to be extremely beneficial. A project can learn and benefit from the experience of previous projects even before it is started. They can be improved over the time through the experience of many different projects

CMMI also gives a holistic approach, it does not concentrate on specific parts of business, and it treats organization as a coherent whole and tries to addresses all problems most IT organizations are struggling with. CMMI defines 22 Process Areas, 54 Goals and 185 Practices. Getting familiar with them can help realize how many important processes and practices are skipped by Scrum teams. CMMI Maturity Levels also help define an order in which missing processes should be introduced. There are years of experience behind CMMI model; it was implemented in hundreds of organizations. It gives its authors unique experience and knowledge which allows deciding which processes are key to the success of the project and which may be implemented later, because the Return of Investment (ROI) would not be so significant. Removing impediments located within Maturity Level 3 Process Area will probably have higher ROI than removing impediments from Process Areas defined on Maturity Levels 4 and 5.

Another important advantage of using some of CMMI practices in Agile environment is the possibility to propagate and improve good solutions over the distance of time. Scrum is mostly oriented at the team and project level, whereas CMMI provides organizational-level infrastructure and mechanism to promote reliable solutions in the whole organizations. Good practices can be normalized, shared and improved among different teams and departments. CMMI also helps to define and standardize definitions of processes which are implemented in an organization. It helps preserve information and knowledge over the time and improve processes. When knowledge is written down and standardized also on boarding processes for new associates, it is much more efficient.

What is important, CMMI can help standardize not only CMMI practices, but also practices derived from Agile and Lean methodologies. Scrum organizations perform regularly same activities for different clients and projects. They carry the same meetings (e.g., Daily Stand-ups, Sprint Planning, Sprint Retrospective and Sprint Review Meetings) and use the same artefacts (e.g., Sprint and Product Backlogs, Burn Down Charts). An organization which adopted Scrum can benefit from formalizing Scrum practices. When organizations use rigorous, well defined and standard processes, it is easier to retain them during the time of stress. This can help the team to stick to their standards when there is pressure to cut corners. This refers to practices and tools derived from many different sources: CMMI, Scrum, XP and Lean.

Such formalism and discipline can also help to deploy Agile methods in large development environments, not only in individual Teams.

4. Conclusions and future work

Both CMMI and Scrum have their shortcomings. The analysis of existing projects shows that neither of them guarantees success of the project. The authors of this paper believe that the use of both approaches in parallel within the same organization or project may increase the probability of project success. CMMI and Scrum can mutually neutralize their limitations and bring software development process to a higher level, impossible to achieve when they are used separately.

In future research the authors want to propose a new approach to managing software development projects. This new approach, based on CMMI and Scrum, assumes that for each newly started project unique management process will be created. This process will be tailored to unique project needs.

Drawing on project factors such as type of project, duration, budget, size of the project team etc., proper CMMI process areas will be chosen. Afterwards, these process areas will be implemented using agile techniques.

References

1. Bass, L., Clements, P., & Kazman, R. (2003). *Software Architecture in Practice*. Boston: Addison-Wesley.
2. Beck, K., & Andres, C. (2004). *Extreme Programming Explained: Embrace Change* (2nd edition). Extreme Programming Explained: Embrace Change.
3. Bergstom, S., & Reberg, L. (2003). *Adopting the Rational Unified Process: Success with the RUP*. Boston: Addison-Wesley Professional.
4. Chrissis, M.B.,Konrad, M., & Shrum, S. (2011). CMMI for Development.
5. Dalton, J. (2011). CMMI and Agile - Partners in Driving Radical Change in Engineering. CMMI and Agile - Partners in Driving Radical Change in Engineering.
6. Drucker, P. (1957). *Landmarks of Tomorrow. Landmarks of Tomorrow*. NY: Harper & Row.
7. Griffin, R. W. (2012). *Management. Principles and practices* (10th edition). Insitute CMMI. Published Appraisal Results.
8. Kruchten, P. (2004). *The rational unified process : an introduction*. Boston: Addison-Wesley.
9. McMahon, P. E. (2010). Integrating CMMI and Agile Development: Case Studies and Proven Techniques for Faster Performance Improvement. Boston: Addison-Wesley Professional.
10. Parth, F., & Snyder, C. (2007). *Introduction to IT Project Management, Management Concepts*. Vienna.
11. PMI (2008). *A Guide to the Project Management Body of Knowledge* (4th edition). USA.
12. Philips, J. (2007). *IT Project management*. Helion.
13. PRINCE2® (2009). Managing Successful Projects with PRINCE2®.
14. Schwaber, K., & Beedle, M. (2002). *Agile Software Development with Scrum*. Prentice Hall.
15. Schwaber, K., & Sutherland, J. (2011). SCRUM Guide.
16. Schwalbe, K. (2010). *Information technology Project Management, Course Technology*. Boston.
17. Sutherland, J., Carsten Ruseng Jakobsen, C.R., & Johnson, K. (2010). Scrum and CMMI Level 5: The Magic Potion for Code Warriors.
18. Wiegers, K. E. (2003). Software requirements (2nd edition). Washington: Microsoft Press.

A. Ziółkowski, T. Deręgowski

Hibridinė prieiga valdant projektus: Galimybių brandos modelio integracijos derinimas su „lanksčiomis” praktikomis

Santrauka

Straipsnis pristato hibridinę programinės įrangos kūrimo projektų prieigą. Ji yra pagrįsta dviem programinės įrangos kūrimo metodologijomis, kurios paprastai traktuojamos kaip nesuderinamos: „sunkiosios” metodologijos, pvz., PMI, PRINCE2 ir RUP, ir „lengvosios” metodologijos, pvz., Scrum, Kanban, XP ir Lean.

„Sunkiosios” metodologijos yra nurodomojo pobūdžio, joms būdinga daug taisyklių, kurių reikia laikytis, daug apibrėžtų vaidmenų ir įrankių. Jos numato, kad formalus ir detalus procesas, išsamiai apibūdinantis visus programinės įrangos kūrimo projekto aspektus, yra esminis projekto sėkmės garantas. Projekto sėkmė yra suprantama kaip projekto pradžioje numatyto produkto pateikimas suplanuoto biudžeto ir laiko rėmų ribose.

„Lengvosios” metodologijos yra adaptyvios ir turi mažiau taisyklių, jos pagrįstos sinergija ir komandų saviorganizacija. Jos numato, kad projekto sėkmę lemia nuolatinis projekto metu vykstantis pokyčių adaptavimas kliento reikalavimams. Nėra svarbu projektą baigti laiku, laikantis biudžeto ir numatytų ribų. Projektas yra sėkmingas tada, kai jam pasibaigus suinteresuoti subjektai gauną tą produktą, kurio jiems reikėjo.

Straipsnyje pristatoma koncepcija taip pat pagrįsta gerai žinomomis proceso tobulinimo programomis. Jos dažnai pradedamos vykdyti programinę įrangą kuriančiose organizacijose, siekiant optimizuoti ir efektyvinti fundamentalius organizacijos procesus. Šios programos apibrėžia procesus, kurie turėtų būti įdiegti sėkmingoje organizacijoje, bet neapibrėžia diegimo būdo. Tokių standartų pavyzdžiai yra CMMI, ITIL ir TOGAF.

Yra daug projektų, kurie sėkmingai taikė tradicinę projektų valdymo prieigą (pvz., CMMI), pavyzdžių. Taip pat yra daug pavyzdžių projektų, kurie buvo sėkmingai vykdyti taikant lanksčiąsias technikas. Remiantis situacinio požiūrio vadybos principais, tinkamiausią prieigą nulemia konkretaus projekto specifika, jo entropija ir kompleksiškumas.

Straipsnio autoriai siūlo kurti naują, adaptyvų ir situacinį IT organizacijos modelį, kuris numato dinamišką unikalaus vadybinio proceso kūrimą kiekvienam naujam projektui. Ši nauja prieiga turėtų remtis ir tradicinių, ir lanksčiųjų IT standartų įvairove. Tokiu būdu būtų pasiekiamas sinerginis efektas.

Sprendimai, kaip įgyvendinti kiekvieną konkretaus programinės įrangos kūrimo projekto dalį, turėtų būti paremti unikalia projekto specifika. Reikia atsižvelgti į tokius unikalius projekto parametrus kaip klientų tipas, kūrimo komandos dydis, patirtis tam tikros technologijos srityje, produkto tipas.

Straipsnyje autoriai taip pat analizuoja dvi pasirinktas prieigas: Scrum (atstovauja „lengvąsias metodologijas”) ir CMMI (atstovauja „sunkiąsias“ metodologijas). Tam, kad būtų galima geriau suprasti hibridinę prieigą, straipsnyje pateikiamos trumpos abiejų metodologijų charakteristikos, jų panašumai ir skirtumai. Scrum kaip pavyzdys buvo pasirinktas todėl, kad tai populiariausiai ir geriausiai žinoma lanksčioji metodologija. Taip pat svarbu buvo tai, kad dauguma Scrum elementų, tokių kaip iteracijos, nuoseklus įrangos kūrimas, savivaldžios komandos ir prisitaikymas prie besikeičiančių reikalavimų yra būdingos kitoms lanksčiosioms metodologijoms.

CMMI yra proceso tobulinimo metodas, kuris leidžia integruoti visus organizacijoje egzistuojančius procesus ir procedūras bei identifikuoti potencialius trūkumus. CMMI modelis buvo sukurtas JAV Gynybos departamento ir yra dažniausiai taikomas kompanijose, kurių veikla susijusi su karinėmis struktūromis, valstybinėse agentūrose ar didelėse korporacijose. Tokio tipo organizacijose svarbiausi veiksniai yra konfidencialumas, saugumas, stabilumas, bet ne kaina ar gebėjimas prisitaikyti prie besikeičiančių klientų poreikių. CMMI paprastai taikomas didelės rizikos, ypač susijusios su kaina, srityse.

CMMI identifikavimas su „sunkiosiomis“ metodologijomis yra tam tikras supaprastinimas. CMMI apibrėžia, kurie procesai turi būti įgyvendinti ir kodėl, bet nenumato, kaip jie turėtų būti įgyvendinami. Todėl iteratyvi prieiga ir „krioklio“ gyvavimo ciklo modelis visiškai neprieštarauja CMMI. Tačiau CMMI buvo pasirinkta kaip „sunkioji“ metodologija, nes dažniausiai ji įgyvendinama kaip „krioklio“ procesas.

Scrum yra viena populiariausių ir geriausiai žinomų lanksčiųjų metodologijų. Daug Scrum elementų, tokių kaip iteracijos, nuoseklus įrangos kūrimas, savivaldžios komandos ir prisitaikymas prie besikeičiančių reikalavimų yra būdingos kitoms lanksčiosioms metodologijoms. Scrum bruožai turi bendrą tikslą: padaryti bendradarbiavimo su klientu procesą sklandesniu. Lanksčiosios metodologijos užsakovą traktuoja kaip partnerį, komandos narį. Jis aktyviai dalyvauja kūrimo procese ir gali jį veikti viso projekto metu.

Pristatę CMMI modelį ir Scrum, autoriai identifikuoja skirtumus tarp šių prieigų. Paprastai CMMI ir Scrum traktuojami kaip nesuderinami. Autorių žinios ir patirtis rodo, kad tai nėra tiesa. Ir CMMI, ir Scrum pasižymi nuosekliais tikslais: aukštesnė kokybė, laimingesnis klientas, trumpesnis kelias į rinką. Abi prieigos gali būti labai efektyvios, atnešti naudą projektui ir organizacijos veiklai.

Toliau straipsnyje analizuojama, kaip galima ištaisyti CMMI ir Scrum trūkumus derinant abi prieigas. Autoriai pastebi, kad CMMI ir lanksčiosios metodologijos derinimas viename procese gali ir padėti tenkinti kliento poreikius, ir sukurti procesą, kuris tuo pačiu metu yra ir brandus, ir dinamiškas. Tokiu būdu galime pasiekti sinergijos efekto ir kurti lankstų procesą ant tvirto CMMI pagrindo, kurti modelių ir metodų derinį su atrinktomis CMMI ir Scrum technikomis, siekiant išspręsti specifines problemas.

Aiškindami, kaip šios dvi prieigos gali būti derinamos, autoriai pirmiausiai nurodo, kaip Scrum gali būti papildytas geriausiomis CMMI praktikomis. Pastebima, kad dabartiniai projektai priklauso nuo specializuotų, techninių žinių turinčių inžinierių. Netgi techninį išsilavinimą turintis projektų vadybininkas niekada neturės tiek žinių ir patirties, kiek jo vadovaujamos komandos nariai ir negalės efektyviai vertinti techninių užduočių.

CMMI neapibrėžia mechanizmų ir įrankių, naudingų žinių darbuotojų savivaldai ir padedančių komunikuoti pažangą vadovybei. Tokius įrankius galima rasti lanksčiųjų praktikų kontekste. Scrum apibrėžia savivaldžias komandas, kuriose komandos nariai patys pasiskiria užduotis, kontroliuoja progresą ir reguliariai apie jį

komunikuoja. Scrum tai pat suteikia adekvačius įrankius, leidžiančius savivaldžių komandų koncepciją įgyvendinti.

Autoriai straipsnyje taip pat analizuoja Scrum trūkumų įveikimo, taikant CMMI, galimybes. Daugelyje organizacijų, kurios taiko Scrum, programinės įrangos kūrimo procesas yra reaktyvus. Tokios organizacijos reaguoja į pastarojo laikotarpio klientų poreikius, bet gana dažnai neįvertina ilgalaikės perspektyvos.

Labai svarbu tai, kad CMMI gali prisidėti standartizuojant ne tik CMMI praktikas, bet ir iš Agile bei Lean metodologijų kilusias praktikas. Scrum taikančios organizacijos nuolat atlieka tas pačias veiklas, tik jos skirtos skirtingiems klientams ir projektams. Jos organizuoja tuos pačius susitikimus ir naudojasi tais pačiais artefaktais. Kai organizacija taiko tikslius, gerai apibrėžtus ir standartinius procesus, yra lengviau juos išlaikyti sunkmečio sąlygomis. Jie gali komandai padėti laikytis standartų net ir tada, kai patiriamas spaudimas taikyti apribojimus. Tai galioja praktikoms ir įrankiams, susiformavusiems veikiant įvairiems šaltiniams: CMMI, Scrum, XP ir Lean.

Apibendrinant atkreipiamas dėmesys, kad ir CMMI, ir Scrum turi trūkumų. Egzistuojančių projektų analizė rodo, kad nei viena prieiga negarantuoja projekto sėkmės. Autorių įsitikinimu, lygiagretus abiejų prieigų taikymas toje pačioje organizacijoje ar projekte galėtų padidinti projekto sėkmės galimybę. CMMI ir Scrum galėtų viena kitą papildyti ir patobulinti įrangos kūrimo proceso lygmenį. To neįmanoma pasiekti, kai šios prieigos taikomos atskirai. Autoriai taip pat pastebi, kad tolimesniuose tyrimuose reikėtų pasiūlyti naują požiūrį į programinės įrangos kūrimo projektų vadybą. Toks naujas požiūris, pagrįstas CMMI ir Scrum, numato, kad kiekvienam naujam projektui bus kuriamas unikalus valdymo procesas, pritaikytas unikaliems projekto poreikiams. Remiantis tokiais projekto veiksniais, kaip tipas, trukmė, biudžetas, projekto komandos dydis ir kt., bus parenkamos adekvačios CMMI proceso sritys, kurios vėliau bus įgyvendinamos taikant lanksčiąsias technikas.

Reikšminiai žodžiai: projektų valdymas, standartai, sunkiosios ir lengvosios metodologijos, CMMI, Agile, Scrum.

First received: June, 2014
Accepted for publication: September, 2014