Software Product Management and Agile Software Development: Conflicts and Solutions

Hans-Bernd Kittlaus

Abstract

Agile software development has been established over the last 15 years as a popular development approach. In a time when speed of change is of utmost importance, agile approaches are often the most appropriate roads to success. They do not only change the way development is performed, but they also impact other parties involved in development projects, in particular the software product manager. Software companies are faced with the question how software product management and agile development can work together in an optimal way. Who is responsible for requirements? Is the software product manager automatically the designated "product owner" (Scrum)? Or is "product owner" a new and separate role? Does he/she replace the software product manager?

The Software Product Management Framework which has been developed by the *International Software Product Management Association* (ISPMA e.V., www.ispma.org) provides orientation. It can be used as a helpful tool to make the change process towards agile development successful.

1 Introduction

Agile – what a wonderful word! Everybody wants to be agile. Marketing people rejoice! Amazing that some presumably nerdy software people came up with the idea to use that term in relation to a new approach for software development and set the fundamentals of that new approach in stone with the "Agile Manifesto"

Special thanks to Gerald Heller for inspiring discussions in the preparation of our joint invited talk on "Software Product Management and Agility" at the International Workshop on Software Product Management (IWSPM) 2012 in Essen, Germany.

H.-B. Kittlaus
InnoTivum Consulting, Rheinbreitbach, Germany
e-mail: hbk@innotivum.de

(Beck et al. 2001). Over the last 15 years this approach has changed the landscape of software development methodology in a significant way.

Agile – as opposed to slow, bureaucratic, old-fashioned, complicated, hindering. Both the Agile Manifesto and the Scrum Guide (Schwaber and Sutherland 2011) are clearly focused on software development only. But it must have been too tempting to extend the scope of that word to other areas. Roman Pichler uses it in “Agile Product Management with Scrum” (Pichler 2010) which deals with the role of “Product Owner” in Scrum without explaining that the spectrum of activities and responsibilities of a product manager is much larger than this product owner role. Dean Leffingwell writes about “Agile Software Requirements” (Leffingwell 2011) – oops, not just people, process, or methodology are agile, the requirements themselves are. This semantic mismatch should not keep anybody from reading the book since it provides a rather balanced approach how Software Product Management and agile methodologies can be combined. The Requirements Engineering (RE) community wants to be agile as well (see Rainer Grau’s article in this book on “Agile RE”).

Agile Software Product Management – from a marketing perspective, we should use that as the title of this article. But we do not – for several reasons. First of all, a software product manager being responsible for the economic success of a product has always had to be “agile” if he or she wanted to be successful. That is nothing new, but has been part of the job description long before the term “agile” was applied to a software development approach (see Kittlaus and Clough 2009). Secondly, the success of agile approaches to software development does not mean that one size fits all. There can still be development projects where due to contents, people and other conditions different methodological approaches like iterative development or even the good old waterfall model may be appropriate (see Figs. 1 and 2). A mature software development organization should be able to choose the optimal method for each individual project, and the software product manager should be able to cooperate with the project teams whatever the chosen development method is. Thirdly, the agile approaches were originally intended for and focused on software development. Then the agile community, in particular Jeff Sutherland and Ken Schwaber started to apply the agile ideas to enterprises (Schwaber 2007), even outside of the IT industry, be it in church (Sutherland et al. 2009) or in sales (de Waard et al. 2011). There are certainly concepts in agile approaches like Scrum that can be helpful for other organizational units of a company or other industries. However, the idea to fundamentally change the way a whole enterprise is run modeled after a software development methodology seems to be rather challenging from a marketing perspective, given the reputation that a lot of corporate IT organizations enjoy in their respective corporations.

So for the purpose of this article let us restrict “Agile” to software development and analyze how software product management can cooperate and interact with an agile software development project. The history and status of software product management are described in this book by Samuel Fricker (2012), so we will not repeat that here. A short history of agile development approaches and a description of the key concepts of Scrum as the market leader can be found in chapter 2. In

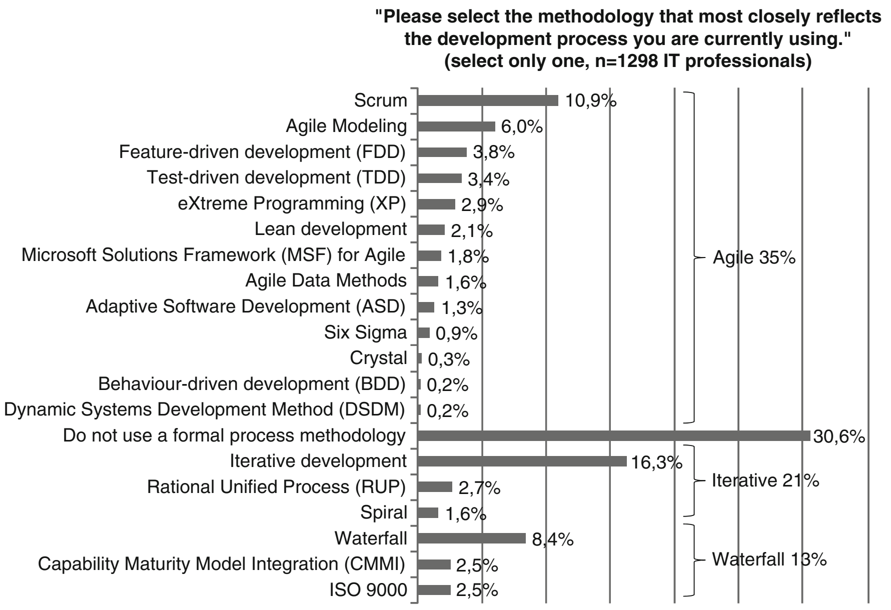


Fig. 1 Agile methodologies (Source: Forrester/Dr. Dobb's Global Developer Technographics® Survey, Q3 2009)

Percentage of a company's development projects using agile methodologies	0 - 25	26 - 50	51 - 75	76 - 100
Percentage of respondents	39	21	12	27

Fig. 2 Percentage of companies' projects using agile (VersionOne 2011)

chapter 3 we will analyze the areas of conflict between Software Product Management and Scrum and show how to solve these conflicts and cooperate and interact in a productive way. Chapter 4 looks at the management implications of chapter 3's findings.

2 Agile Development

2.1 Short History

Ever since software started to be created in the 1950s, it has had an unprecedented track record of amazing impact on business and society, of being the source of incredible wealth and disastrous failure, of triumphant success and deep frustration.

The more important software became from a business perspective, the stronger became the desire to make the process of creating software more manageable, more reliable, more "engineering"-like or more manufacturing-like. So it reflected more wishful thinking than reality when the term "software engineering" was coined in 1968 or the term "software factory" in the 1980s (Kittlaus 2003). And there are good arguments why these terms do still not describe reality (Davis 2011; White and Simons 2002).

Nevertheless, in order to improve a rather unsatisfying situation, the industry turned more and more to methodology based on practical experience. For software development, the waterfall model had been dominant since the 1970s which is a phase model in which one phase needs to be finished before the next can begin. Bigger real world software development projects have never really worked like that, but that model found its correspondence in project management methods which started to be standardized in the 1980s. Examples are PMI or PRINCE2 which come with training, certification and consulting. The move to methodologies was a push from management and consultants, not a pull from developers who typically viewed them as restrictions of their freedom, their creativity and their productivity. The next wave of software development methodology was iterative development which took into account that cutting a piece of work into smaller chunks which could be developed one after the other increased the probability of success and gave management a better feeling for progress. Fowler refers to all these approaches as "engineering methodologies" (or plan-driven methodologies). In the mid-1990s agile methods started to become popular as Martin Fowler describes in (Fowler (2005): "Engineering methodologies have been around for a long time. They've not been noticeable for being terribly successful. They are even less noted for being popular. The most frequent criticism of these methodologies is that they are bureaucratic. There's so much stuff to do to follow the methodology that the whole pace of development slows down."

To some degree, agile methodologies can be seen as a reaction to these engineering methodologies, providing "just enough" process. That means a smaller amount of documentation and more code-orientation. Fowler sees deeper differences (Fowler 2005):

- "*Agile methods are adaptive rather than predictive*. Engineering methods tend to try to plan out a large part of the software process in great detail for a long span of time, this works well until things change. So their nature is to resist change. The agile methods, however, welcome change. They try to be processes that adapt and thrive on change, even to the point of changing themselves.
- *Agile methods are people-oriented rather than process-oriented*. The goal of engineering methods is to define a process that will work well whoever happens to be using it. Agile methods assert that no process will ever make up the skill of the development team, so the role of a process is to support the development team in their work."

The term “agile” was agreed upon in a workshop in 2001 that was attended by 17 method gurus including Fowler (Fowler 2005). It resulted in the *Manifesto for Agile Software Development* (Beck et al. 2001) which gained a lot of attention and is worth citing here in full:

“We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.”

Advocates of the established methodologies, be it in software development or in project management, considered this manifesto as a declaration of war. Some tried to associate “agile” with the old hacking, i.e. software development without plan or documentation, but to no avail. The community of software developers striked back, the term stuck, and agile approaches have become more and more popular over time.

In the late 1990s eXtreme Programming (XP) developed by Kent Beck and others (Beck 1999, 2004) got the most attention of all agile approaches. It is not only a framework and philosophy, but gives very practical advice in the form of concrete techniques, so called practices. Crystal was developed by Alistair Cockburn and is more light-weight. It comes in a number of variations for different sizes of projects, but not all variations are as properly documented as Crystal Clear (Cockburn 2004). There have been a number of other approaches, the most popular of which has been Scrum developed by Jeff Sutherland and Ken Schwaber (Schwaber 2004; Schwaber and Beedle 2001).

There are some statistics available that make quantitative statements about the adoption of agile methods in general and Scrum in particular. Forrester Research (West and Grant 2010) published Fig. 1.

Based on this research conducted in 2009 agile methodologies had a market share of 35 % with Scrum being the agile market leader at 10.9 %.

The results of VersionOne’s 2011 State of Agile Survey (VersionOne 2011) show even higher adoption rates (Fig. 2 and 3).

Since VersionOne is a vendor of tools for agile development, it is not clear if the 6,042 international participants in the study are really representative of the total worldwide software development community. So the numbers regarding the adoption rate may be a bit too high. Even with these numbers, it is obvious that the majority of companies has not moved to agile development fully, i.e. software product management has to cooperate with both agile and non-agile development teams. Given Scrum’s high market share of 52 %, or 66 % when Scrum Hybrids are included, for agile development this article focuses on Scrum (Fig. 3).

Fig. 3 Agile methodology most closely followed in agile projects (VersionOne 2011)

Agile Methodology	**Market Share (in %)**
Scrum	52
Scrum / XP Hybrid	14
Custom Hybrid	9
Kanban	3
Scumban	3
Feature-Driven Development	2
Extreme Programming XP	2
Lean	2
Other	5
Don’t Know	8

2.2 Scrum: Key Concepts

Scrum is not a fully elaborated method, but rather a framework based on a philosophy that values self-organization and the individual skills and abilities of the team members highly. The guiding document for Scrum is the Scrum Guide published by the creators of Scrum, Jeff Sutherland and Ken Schwaber. In its 2011 edition it has just 13 pages with contents (Schwaber and Sutherland 2011). Compared to the 2010 edition, the authors removed and changed some concepts. So Scrum is a moving target which may contribute to its success. Even though Schwaber and Sutherland state that “A common language referring to the process must be shared by all participants.”, the Scrum Guide does not define fundamental terms like “product” or “release”.

A project is organized in iterations called Sprints that must not last more than a month each. Other Scrum Events are Sprint Planning Meeting, Daily Scrum, Sprint Review, and Sprint Retrospective.

Additional key elements of Scrum are the following roles in a so-called Scrum Team:

Product Owner

- Responsible for maximizing the value of the product and the work of the Development Team.

- The sole person responsible and accountable for managing the Product Backlog and deciding what the Development Team works on. Work can be delegated to Development Team.
- One person that is respected by the entire organization, not a committee.
- With regard to the Product Owner role the Scrum Guide says "How this is done may vary widely across organizations, Scrum Teams, and individuals" (Schwaber and Sutherland 2011, p. 5).

Development Team

- Responsible for delivering potentially shippable product increments at the end of each Sprint.
- 3–7 people with cross-functional skills who do the actual work.
- Self-organizing.

Scrum Master

- Responsible for ensuring Scrum is understood and enacted.
- Servant-leader for the Scrum Team.
- Protects the Development Team and keeps it focused on the tasks at hand.

The Scrum Guide lists a number of relevant artefacts:

Product Backlog

- Single source of requirements for any changes to be made to the product.
- Lists all features, functions, requirements, enhancements, and fixes that constitute the changes to be made to the product in future releases.
- Product Backlog items have the attributes of a description, order, and estimate.
- Dynamic and evolving
- Grooming, i.e. the act of adding detail, estimates, and order to items in the Product Backlog, is an ongoing process in which the Product Owner and the Development Team collaborate. Estimates are only done by the Development Team.

Sprint Backlog

- Set of Product Backlog items selected for the Sprint plus a plan for delivering; forecast by the Development Team about what functionality will be in the next Increment and the work needed to deliver that functionality.
- Owned and updated by the Development Team.

Increment

- The sum of all the Product Backlog items completed during a Sprint and all previous Sprints.
- Product Owner responsible for the decision if increment is released.

The Scrum Guide states explicitly that Scrum does not define a process or a technique (Schwaber and Sutherland 2011).

3 Areas of Conflict and Solutions

Given the rather rudimentary specification of Scrum which has been changing over time, there are different interpretations and different views on how Scrum can or should be positioned and implemented in an organization. In contrast to traditional development methods, Scrum demands changes not only in development, but also in other parts of the enterprise. The interfaces of the Scrum Team to the rest of the enterprise are embodied in the roles of “Product Owner” and “Scrum Master” which are new with Scrum.

3.1 The Naming

The product owner role is central to Scrum, and with the success of Scrum it has found wide-spread use. It had its origins in the start phase of Scrum when the focus was only on a development project and the implicit understanding of the term “product” was “that what was produced in the development project”. Up to this day, the term “product” has not been explicitly defined in Scrum, but its meaning has shifted towards a broader understanding that is more in line with ISPMA’s definitions (ISPMA 2012b):

- A product is a combination of goods and services, which a supplier/development organization combines in support of its commercial interests to transfer defined rights to a customer.
- A software product is one whose primary component is software.

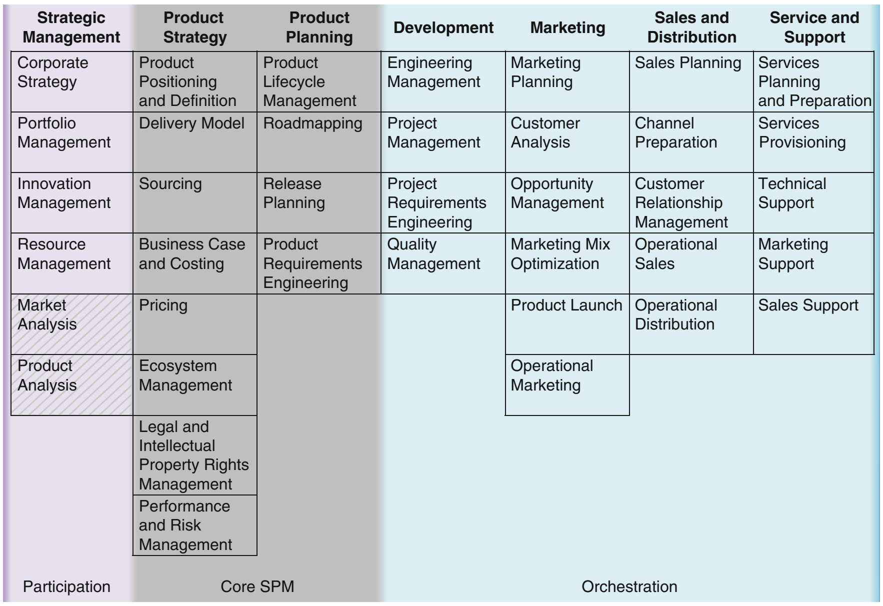
In a lot of software product companies, the term “product owner” is used for a business executive who has the full P&L responsibility for a product. There are cases where the software product manager is called “product owner”. Both situations are in conflict with the Scrum definition of the term. So we suggest that in those environments a different term is used for the Scrum role, e.g. business systems analyst or requirements analyst (see Leffingwell 2011, p. 206).

For the remainder of this article, we use the Scrum role name “product owner”.

3.2 The Roles of Product Owner and Software Product Manager

While the Scrum Master is supposed to shield off the Development Team from the outside world, the Product Owner is to represent the outside world within the Scrum Team. Since the Product Owner is supposed to be an individual, not a group or a committee, this is a daunting task.

Practical experiences and reports in Scrum-related blogs show that these requirements towards the Product Owner can very often not be fulfilled. Schwaber writes in (Schwaber 2007, p. 85): “Until recently, I viewed this relationship (between Product Management/Customer and the Development Team) as one of many changes in a Scrum adoption. I now view it as the most critical change, the lynchpin of the adoption.” For Schwaber it goes without saying that the Product Owner and the Product Manager are the same. In his rather drastic way of phrasing, Schwaber says (Schwaber 2007, p. 83): “Almost all the product management and development work is done in a hierarchy of Scrum teams. Unless remaining staff and managers have other solid work to do, their idle hands are the devil’s workshop. They interfere with the Scrum teams.” So the Product Owner does the product



Strategic Management	Product Strategy	Product Planning	Development	Marketing	Sales and Distribution	Service and Support
Corporate Strategy	Product Positioning and Definition	Product Lifecycle Management	Engineering Management	Marketing Planning	Sales Planning	Services Planning and Preparation
Portfolio Management	Delivery Model	Roadmapping	Project Management	Customer Analysis	Channel Preparation	Services Provisioning
Innovation Management	Sourcing	Release Planning	Project Requirements Engineering	Opportunity Management	Customer Relationship Management	Technical Support
Resource Management	Business Case and Costing	Product Requirements Engineering	Quality Management	Marketing Mix Optimization	Operational Sales	Marketing Support
Market Analysis	Pricing			Product Launch	Operational Distribution	Sales Support
Product Analysis	Ecosystem Management			Operational Marketing		
	Legal and Intellectual Property Rights Management					
	Performance and Risk Management					
Participation	Core SPM		Orchestration			

Fig. 4 ISPMA software product management reference framework V.1.1 (ISPMA 2012a)

management work, and if an employee is not part of a Scrum team, he/she is not only superfluous, but dangerous. Roman Pichler is not as drastic, but works from the same assumption in (Pichler 2010), i.e. the Product Owner does the product management.

In Fig. 4 the ISPMA's SPM Reference Framework is illustrated. It is structured in the following way:

– The horizontal structure (columns) is based on the functional areas of a software organization.
– Vertically, i.e. within the columns, the structure is based on a top-down approach, i.e. from more strategic and long-term to more operational and short-term.
– There is an additional overlay structure with "Core SPM" (grey shading), "Participation" and "Orchestration". For Market Analysis and Product Analysis in the Strategic Management column the responsibility is typically with corporate functions in larger companies with the product manager participating, in smaller companies the product manager may be responsible. In any case, getting reliable information on market and product on a frequent basis is part of the core SPM responsibilities.

A more detailed explanation of the framework and its elements can be found in ISPMA (2012b).

When looking at the detailed description of the tasks a Product Owner is responsible for, some of them can be found in the SPM framework (Fig. 4), in particular product vision and requirements engineering. However, there are so many additional tasks listed in that framework that are not part of the Scrum

Product Owner role. And when a product manager covers all the tasks in the framework, he is typically more than busy and not able to assume additional responsibilities that Scrum imposes. That is the line of thought that Dean Leffingwell is following in Leffingwell (2011). He says "Given this (the product manager's) set of responsibilities, it is clear that – even with a staff of competent product owners – product management remains an important function in agile development . . ." (Leffingwell 2011, p. 280).

So in a small organization with just one software product manager and one Scrum team, the product manager will often assume the product owner role in the Scrum team. This is also described in case studies in Vlaanderen et al. (2012). Vlaanderen et al. (2011) describes the "Agile Requirements Refinery", an approach how software product management can apply Scrum principles to its own work on requirements. Vlaanderen et al. (2012) contains a case study in which product management assumes the product owner role in up to seven Scrum teams, but does not give details about the number of product managers and the impact this has on the rest of their responsibilities. Our own experience is more in line with Leffingwell (2011, p. 205) that that approach does not scale up, i.e. as soon as the organization is bigger and there are multiple Scrum teams working on the same product, the product manager cannot and should not assume the product owner role in all these teams. Plus product managers may neither be willing nor able to work on a very technical level close to development due to their individual backgrounds.

The solution is a split of responsibilities between product manager and product owner that needs to be clearly defined. The ISPMA SPM Framework (Fig. 4) turns out to be very helpful in determining this definition in detail. In general, the product owner is closer to development, technology and the project aspects of the product, i.e. in the Development column of the framework. The product manager is closer to the business, the customers, the life cycle aspects of the product, i.e. in the Product Planning and Product Strategy columns of the framework. This is very much in line with Leffingwell's view (Leffingwell 2011, p. 288).

Basically the product owner is a member of the Scrum team and the development organization, with a strong dotted line to Software Product Management, i.e. it is some kind of matrix organization. The product owner is responsible for (see Leffingwell 2011, pp. 51 and 207–208):

- Managing the backlog (project requirements engineering),
- Performing just-in-time story elaboration (detailed requirement specifications), and accepting new stories,
- Participating in sprint planning meetings and progress reviews,
- Driving the iteration,
- Collaborating with product management, e.g. on release planning.

The product manager has to adapt his/her activities in a number of aspects (see also Leffingwell 2011, p. 283 ff.) if development utilizes agile methodologies:

- Product Requirements Engineering: Analysis and specification more high-level since details will be determined in the product owner's story elaboration; tight and ongoing cooperation with product owner regarding synchronization of product and project RE. Vlaanderen et al. (2011) describes how Scrum principles can be applied on the SPM side.

- Release Planning: More flexibility regarding changes during development phase, i.e. contents and prioritization of requirements and addition of new requirements; release dates more reliable, scope more flexible which impacts expectation management (see also Leffingwell 2011, pp. 299 ff.). In this book, Theuns et al. (2012) describe a case study on the impact of the adoption of Scrum on release planning in the case that the product manager assumes the product owner role.
- Roadmapping: Higher change rate due to increased flexibility in release planning.

With this split of responsibilities, the roles of product owner and software product manager can be adequately defined and positioned so that productive cooperation is facilitated. Company-specific details can be defined based on the SPM Framework. The success of an implementation is highly dependent on the availability of people who have the skills and abilities to convincingly fill these positions.

3.3 The Timing Considerations

The agile methodologies, and Scrum in particular, are focused on creating a work environment for the developers that enables high productivity by ensuring a continuous flow of elaborated user stories in a sequence governed by value assessments. This puts a lot of pressure on the product owner who is responsible for the timely availability of these user stories. In cases where a software product manager assumes the product owner role, these demands can easily lead to overload situations and/or to neglection of other product management responsibilities. Vlaanderen et al. (2011) proposes the agile requirements refinery as an approach to meet these demands by applying Scrum principles within software product management. The corresponding Sprint cycles can be overlapping.

When the product owner role is not assumed by software product management, the work of the software product manager is less directly triggered by the Scrum rhythm. This enables the product manager to synchronize his work more with the frequencies of other important processes that require his involvement, like corporate planning processes (e.g. portfolio management, marketing plan, sales plan) or his own processes like roadmapping. In short, he can focus more on the important than the urgent (Kittlaus and Clough 2009, p. 42) while he is still sufficiently involved in project requirements decisions through the dotted line from the product owner to software product management.

4 Management Implications

From a management perspective, agile methodologies like Scrum in software development promise a number of significant advantages like higher productivity and faster reaction to changing requirements. The adoption, however, means a significant change process and takes a longer period of time (see the case study in

Theuns et al. 2012). It requires a champion on the executive level and guidance from experienced consultants. In most companies this is not a black-or-white issue, i.e. there may be development projects that will continue to be better served by more traditional development methodologies.

If Scrum is used in a development project this has significant implications for other organizational units within the company, in particular software product management. These SPM implications result from the relationship between the software product manager and the Scrum product owner roles and are described in 3.2.

The ISPMA SPM Framework (Fig. 4) helps to resolve any conflicts. We suggest using it as a basis for the following steps:

- Analyze as-is situation
- Identify current owners of tasks
- Identify tasks not taken by anyone
- Clarify and communicate definitions of relevant terms across company
- Establish company-wide roles and responsibilities
- Find the optimal balance and cooperation between SPM and agile development teams

As above, this requires a champion on the executive level and guidance from experienced consultants.

Though we do not recommend the adoption of a vanilla Scrum approach to all units of a company, there are a number of elements in Scrum that can be very useful in improving productivity and time-to-market in units other than development, in particular:

- Team approach
  - With small teams (5–9 people)
  - Dedicated not only in terms of mindset, but also in terms of time allocation
  - Leaving room for self-organization, but with some key roles and responsibilities.
- Appreciation of the individual skills and abilities of the team members.
- Organization of work in time-boxed iterations with frequent “success” points (Sprints).
- Team communication structured and organized in a way that enforces sufficient communication and learning without sacrificing productivity (Sprint Planning Meeting, Daily Scrum, Sprint Review, and Sprint Retrospective).

If and how this can be implemented in the other units including SPM needs to be determined by the responsible management. Again, a guided change management process is required.

5 Summary

Scrum as the market leader in agile methodologies for software development projects contains terminology and the role definition of “product owner” including its demands regarding timing which are in conflict with the state of the art of software product management. In this article we have described the conflicts and developed solutions how to deal with these conflicts in a way that enables and ensures productive cooperation. The ISPMA Book of Knowledge, in particular the SPM Framework prove to be very helpful in analyzing a given situation in an organization and define specific solutions in detail. Some elements of agile approaches may also be helpful and applicable in a company's units outside of development in order to improve productivity and time-to-market.

So far there has been very little scientific work and publications on the relationship of Software Product Management and agile methodologies. Progress has primarily been driven by consultants and companies adopting agile methodologies. There is a lot of room for research in the areas of software product management, software development methodology, and economic sciences.

References

Beck, K. (1999). *eXtreme programming explained – Embrace change*. Upper Saddle River: Addison-Wesley.

Beck, K. (2004). *eXtreme programming explained – Embrace change* (2nd ed.). Upper Saddle River: Addison-Wesley.

Beck, K., et al. (2001). Manifesto for agile software development. http://agilemanifesto.org/

Cockburn, A. (2004). *Crystal clear*. Upper Saddle River: Addison-Wesley.

Davis, M. (2011). Will software engineering ever be engineering? *CACM, 54*(11), 32–34.

de Waard, D., van Solingen, R., & Sutherland, J. (2011). Scrum in sales: How to improve account management and sales processes. *Agile conference 2011,* Salt Lake City.

Fowler, M. (2005). The new methodology. http://martinfowler.com/articles/newMethodology.html

Fricker, S. A. (2012). *Software product management*, in this book.

ISPMA. (2012a). Software product management reference framework V.1.1. www.ispma.org

ISPMA. (2012b). Software product management – Foundation level syllabus V.1.1. www.ispma.org

Kittlaus, H.-B. (2003). Software Engineering und Software Fabrik – vom Nutzen und Schaden der Metapher in der Informatik. In: Informatik-Spektrum 26: pp. 8–12 and 291–292.

Kittlaus, H.-B., & Clough, P. C. (2009). *Software product management and pricing – Key success factors for all software organizations*. Heidelberg: Springer.

Leffingwell, D. (2011). *Agile software requirements*. Upper Saddle River: Addison-Wesley.

Pichler, R. (2010). *Agile product management with scrum*. Upper Saddle River: Addison-Wesley.

Schwaber, K. (2004). *Agile project management with scrum*. Redmond: Microsoft Press.

Schwaber, K. (2007). *The enterprise and scrum*. Redmond: Microsoft Press.

Schwaber, K., & Beedle, M. (2001). *Agile software development with scrum*. Upper Saddle River: Prentice Hall.

Schwaber, K., & Sutherland, J. (2011). The scrum guide. http://www.scrum.org/storage/scrumguides/Scrum_Guide.pdf

Sutherland, A., Sutherland, J., & Hegarty, C. (2009). Scrum in church: Saving the world one team at a time. *Agile conference 2009*, Chicago.

Theuns, M., Vlaanderen, K., & Brinkkemper, S. (2012). *Exploring the relationship between scrum and release planning activities*, in this book.

VersionOne. (2011). State of agile survey 2011. http://www.versionone.com/state_of_agile_development_survey/11/

Vlaanderen, K., Jansen, S., Brinkkemper, S., & Jaspers, E. (2011). The agile requirements refinery: Applying SCRUM principles to software product management. *Information and Software Technology, 53*(2011), 58–70.

Vlaanderen, K., van Steen, P., Brinkkemper, S., & van de Weerd, I. (2012). Growing into agility: Process implementation paths for SCRUM from an SPM perspective. *Proceedings of the 13th International Conference on Product-Focused Software Process Improvement*, Heidelberg/New York, NY: Springer.

West, D., & Grant, T. (2010). *Agile development: Mainstream adoption has changed agility*. Cambridge, MA: Forrester Research.

White, J., & Simons, B. (2002). ACM's position on the licencing of software engineers. *CACM, 45*(11), 91.