Qualitative Study of Successful Agile Software Development Projects

Various studies show that the agile method has become a mainstream methodology for software development. When agile pioneers introduced this approach, they executed very successful projects which lead to the enormous popularity of agile development. With becoming mainstream, less experienced teams started to apply the agile approaches and news about failed agile projects appeared. This raises the question, what it needs to conduct successful agile projects. In a qualitative study we asked IT companies about the essential success factors in their successful agile projects. We found that there was a strong focus on engineering and management best practices. We found that when these practices did not work, mature teams sensed that following a recipe is not sufficient, and they started adapting the agile process to their needs. Applying a sense-making methodology like the Cynefin framework, theoretically explains our observations in the study.

Martin Kropp, Andreas Meier | martin.kropp@fhnw.ch

In recent years, agile methods have become one of the predominant ways for software development as various studies show [1] [3]. This is not only because agile has become hype and any "state-of-the-art" IT company "just does it". The mentioned studies also clearly show improvements over traditional approaches, like faster time-to-market, significantly improved change management, and higher satisfaction with the overall process. However the studies also show, that a significant part of agile projects fail due to various reasons, though they apply most of the recommended good practices for agile development.

Thus the authors wanted to find out, what the essential factors for successful agile projects are. For this they conducted an interview study among several Swiss IT companies about their most successful agile projects. The goal was to get an understanding of what the essential criteria for successful agile projects are.

The first section provides an overview about related work. Then we present the qualitative study, with its central questions and the applied study method, followed by the major study results. In the next section we introduce the theory of Complex Adaptive Systems which helps to understand how successful agile software projects and teams function and work. After that we present and discuss the findings about the critical success factors from our interview study. The paper closes with a final conclusion and an outlook on future work.

Related Work

There are many studies, which show the distribution and success of agile methods for software development. The company Version One executes a yearly study about usage of agile methods and practices [1] in the United States. Reference [3] is a similar bi-yearly study focused on the usage of agile and plan-driven approaches in Switzerland. In [2] Kaltenecker et. al. present the results of an interview study with the focus on leadership in agile projects.

Already in 2001, A. Cockburn and J. Highsmith discussed the still neglected importance of the people factor when introducing agile methodologies [20]. Recent research [21] and [22] goes further and analyses the influence of organizational culture and agile methodology.

In [9] and [5] Snowden et. al. discuss the relation between complex systems and agile software development and present the *Cynefin* framework [4] for agile leadership.

Interview Study

The goal of our study was to show by example how successful agile projects are realized in various kinds of projects and different kinds of companies and branches. We conducted an interview study among eight Swiss IT companies that have adopted agile methods in their software development. The companies were asked to provide a written description in advance of their most successful agile project, including a reasoning why they value the described project as successful – from their point of view and from their customers' point of view.

We conducted semi-structured individual interviews and used a self-developed interview guide. We asked questions about all the three competence levels engineering, management and agile values as formulated in [7]. Since all interviews took place in the German speaking part of Switzerland, all interviews were conducted in German.

The interviewed companies operate nationally, internationally and globally. Table 1 gives an overview of the industry branches covered. The interviews were conducted with a total of nine participants (at one company, we had two interviewees, all male). The participants were most-

Branches	Number of Companies
Product Development	2
Public Service	1
Manufacturing	2
Insurance	1
IT Supplier	2

Table 1: Branches covered in study

ly project leaders, group leaders, or department leaders. The interview duration was between one and two hours. All interviews were conducted in German. All interviews were audio recorded and transcribed later. The transcription facilitated the evaluation of the interviews with the help of a statistical text-analysis tool.

Organizational Level of Agility
In this paper we use the terms *team*, *organization* and *company* to describe on which level Agility was introduced:

- *Team* refers to the software developers, testers, Scrum Masters, Product Owners etc., i.e. the people that are directly involved in the software development. All the teams in the study follow an agile methodology like Scrum.
- *Organization* refers to the team plus other people who are involved in the project, i.e. customers, end-users, and management sponsors. They work together with the team in an agile or non-agile environment. In the former case, the team supports the organization to become agile. In the latter case, the team provides an "interface" to facilitate the communication between the different environments.
- *Company* refers to the enterprise within which the software development project is executed. Currently, most companies are not agile. In our interviews, we had seven non-agile companies where at least one team or organization was agile.

In seven of the eight companies the agile approach was applied either for one or more organizations within the company or on the team level for the project teams. In these companies, agile methodologies were either introduced bottom-up or top-down. That means those teams where typically embedded in a classical, hierarchical organization within its own company. In one company, agile methodology was introduced in the whole company, i.e. was applied also on management level.

Success
1. What makes the project successful from your point of view?
2. What makes the project successful from your customer's point of view?
3. What are the reasons for the success from your point of view?
Engineering
4. How much do the engineering practices (according to [7]) contribute to the success?
5. Which engineering practices do you apply regularly?
Organization/Management
6. How much do the management practices (according to [7]) contribute to the success?
7. Which management practices do you apply regularly?
Culture & Values
8. How much do the cultural aspects / values (according to [7]) contribute to the success?
9. Which cultural aspects / values do you apply regularly?
Improvements
10. What would you change to make the project even more successful?

Table 2: Interview questions translated from German

Interview Questions
As preparation for the interview the interviewees were asked to send in a short description of the selected project, including some basic information about the project like duration, effort, and team size. Additionally, they had to answer the following two questions:

- What makes the project successful from your point of view?
- What makes the project successful from your customer's point of view?

In the one-hour interview (in average), the interviewee had to provide further information about

	P1	P2	P3	P4	P5	P6	P7	P8
Type of Project	Product	In-house	Product	Product	Product	In-House	In-House	Product
Company	IT Solution	Public service	Manu-facturer	Manu-facturer	IT Service Provider	Insurance	IT Solution	Manu-facturer
Method	Scrum	Scrum	Scrum	Scrumban	Scrum	Scrum	Scrum	Scrum
Size	24 PM	30 PM	30 PM	100 PM	30 PM	12 PM	72 PM	> 120 PM
Duration	3 M	10 M	12 M	9 M	10 M	8 M	24 M	>
Team	5 P	10 P	5 P	3 x 4 P	6 P	8 P	5 P	5 (current)

Table 3: Project figures

the company and the project (industry sector, company size, in-house/contract work/product development, main project technology). The questions are listed in Table 2.

Results of the qualitative study

Table 3 gives an overview of the basic project figures. The projects ranged from small projects to intermediate sized projects, and covered in-house projects, embedded software projects and product development. The team size ranged from 5 to 12 persons, the latter being organized as a scrum-of-scrum team.

Success in Agile Projects

The question about why the selected project was considered successful was answered in very many different ways. Among the most popular answers were: very good communication in the team; continuous delivery; delivery on time; very few bugs; satisfied customers; no overtime. When asked for the reasons for the success, the following main aspects were mentioned: all team members were committed; continuous and extensive testing; requirements were very clear; very close communication with and intensive feedback from customers; team workshops for team building.

Engineering Practices for Success

Many different engineering practices are applied by these organizations. Among the most important practices mentioned by almost all teams are continuous testing, at least on unit level (only few apply TDD); continuous integration; and continuous quality control. Some teams use Pair Programming, and some Clean Coding.

Organization Practices for Success

The success criteria on management level which were mentioned most often were the short iterations (typically two weeks) and the self-organization of teams.. More important issues mentioned by others were User Story Grooming during Sprint; close communication with externals; open office. Management support was also argued to be very important by one company. For remote teams daily meetings with visual communication tools like Skype was very important in one company.

In the eight observed projects, many different practices and paradigms were used. Their usage depended on the size and domain of the project but also on the culture of the respective company. We found a small but powerful set of underlying characteristics common in all agile projects. Before summarizing these key characteristics of successful agile projects, we will take a closer look at the theory of complex systems. With the insight of complexity theory, these key characteristics can be better understood and an indication on how and when to apply them in other projects can be deduced.

Complex Adaptive Systems and Agile Competences

In recent years complexity theory has been widely discussed in the scientific community. In this paper we argue that parts of a software development project should be treated as a complex system. More specifically we will point out that some parts of a project are in the *complex domain* and other parts are in the *ordered domain*. It is important to note that a typical software development project is a multi-domain problem. We elaborate what the consequences are and why that matters.

There are different definitions of complex systems. In this paper we use the definition of constraints. Basically there are three different types or ontologies: *chaotic* systems, *complex* systems and *ordered* systems. A system consists of *agents* and the interaction between these agents. Agents interact within the system or on the system. Examples of agents in software development are: the project goal, the project team, customers, the company, the culture within the company, management, competitors, technology, market place, etc. It is important to note that agents are not usually single individuals.

In a chaotic system, the agents are not constrained and thus independent of each other. In an ordered system, the system constrains the agents. These are the two extremes of the spectrum. In between there is the complex system. Such a system slightly constrains the agents. The agents modify the system by their interaction with it and among each other.

A *Complex Adaptive System* (CAS) is a special case of a complex system. Here the constraints and agents are co-evolving. Examples of CAS are the Internet, cyberspace, stock markets, manufacturing businesses, ant colonies, and any human social group-based endeavor. And a software development project is also regarded as a complex adaptive system [9].

Listed below are some of the characteristics of complex adaptive systems [9]:

- A CAS is highly sensitive to small changes. There is the danger that weak signals are easily missed or even dismissed. Small changes can have unpredictable consequences. This is also known as the butterfly effect.
- There is no (or very little) causality and therefore hindsight does not lead to foresight. In other words, a CAS is not causal but dispositional.
- Retrospective coherence, i.e. the fact that a system worked in a certain way in the past does not mean it will work the same way in the future.
- Proximity and connectivity are key

- There are dangers of confusing correlation with causation and simulation with prediction
- "Need to keep your options open". Premature convergence is the main danger of complexity, i.e. converting too quickly to a solution and not to keep multiple possibilities open.

If we look at agile software development projects as complex adaptive systems, there are a number of consequences that follow [9]:

- CAS cannot be reset. Wherever you are, is where you are, e.g. it is not possible to reset a sprint and start over.
- Relationships between agents are more important than the agents themselves. It is more effective to improve the relationships than trying to change every agent. Especially when the agents are groups of individuals.
- Management of ordered and complex systems or domains is important. The team has to know, what kind of domain the project is in and how to respond accordingly (see Cynefin framework below)
- Praxis[1] makes perfect. Agile teams value both theory and practice, e.g. they do not only apply Scrum in practice but they also know the theory of complex systems behind it.

Cynefin

Cynefin [4] is a decision-making framework that recognizes the causal differences that exist between chaotic, complex and ordered systems. It is valuable because it proposes different approaches to decision-making in complex social environments like in software projects.

The name Cynefin, pronounced ku-*nev*-in, is a Welsh word that signifies the multiple factors in our environment and our experience that influence us in ways we can never understand. "*The name seeks to remind us that all human interactions are strongly influenced and frequently determined by the patterns of our multiple experiences, both through the direct influence of personal experience and through collective experience expressed as stories.*" [5]

The Cynefin framework is also a sense-making framework. It is used to help define the system we have to deal with, i.e. lays the problem at hand in the ordered or un-ordered domain. This is important because problems in the ordered domain require very different strategies than the ones in the unordered domain.

Its value is not so much in logical arguments or empirical verifications as in its effect on the sense-making and decision-making capabilities of those who use it. The Cynefin framework has been used for knowledge management, project management, IT-design, strategy making etc. Its purpose is to give decision makers (e.g. the team members,

1 There are different definitions of ‚praxis'. Here ‚praxis' is defined as the combination of theory and practice.

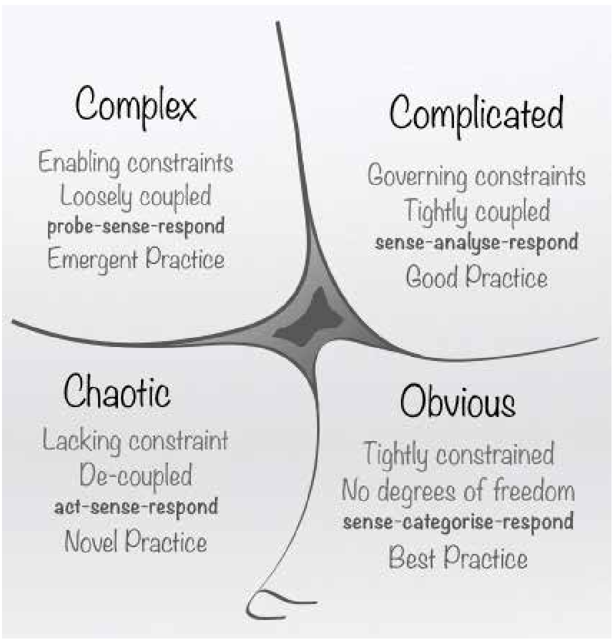

Figure 1: Cynefin domains (Source [11])

agile leaders, management, etc.) powerful new constructs that they can use to make sense of a wide range of unspecified problems. It also helps people to break out of old ways of thinking and to consider intractable problems in new ways [4], [5].

The Cynefin framework does not automatically provide the right answer, but it will help the agile team members to use their skills and experience to look for it in the right place.

The five domains of the Cynefin framework

As can be seen in Fig. 1, the Cynefin framework has five domains, four of which are named, and a fifth, central area, which is the domain of Disorder. The right-hand domains (Obvious, Complicated) are those of order, and the left-hand domains (Complex, Chaotic) are those of unorder.

- *Obvious Systems:* In the obvious (aka simple) domain, cause and effect relationships are obvious and predictable in advance. The causality is self-evident or obvious to any reasonable person. In this domain we apply *best practices*, i.e. established examples of what works in a particular context. The approach is to *sense, categorize* and *respond*.
- *Complicated Systems:* In the complicated domain we have an ordered system where cause and effect relationships exist but they are not self-evident. There is a right answer, however, the answer is not self-evident and requires analysis and/or the application of expert knowledge. With the right expertise, there can be several different ways of doing things in this domain. Applying *good practices*, i.e. a range of examples of what works well in a given context, works well in complicated systems

provided we have the right expertise. The approach is to *sense, analyze* and *respond*.

- *Complex Systems:* In the complex domain we have an un-ordered system where the relationship between cause and effect are only obvious in hindsight. The causality can only be perceived in retrospect and the results are unpredictable. In this domain, we need to create *safe to fail experiments.* And we do not attempt to create fail-safe designs. We cannot solve complex problems with best or good practices alone. While conducting safe to fail experiments, the key is to *dampen* the parts that fail and *amplify* the parts that succeed. In this domain we get emergent order and practice that is often unique. The approach is to *probe, sense,* and *respond*. In this domain we apply *emergent practices*, i.e. new practice and some combination of best and good practices.
- *Chaotic Systems:* In the chaotic domain, no cause and effect relationships can be determined. The approach is to *act, sense,* and *respond.* In order to effectively understand and function in a chaotic system, we must act very quickly to either innovate or stabilize the system and therefore learn from it. In chaotic systems we apply *novel practices*.

Depending on the system or ontology that applies to the situation, we should think and analyze accordingly. One size does not fit all!

- *Disorder:* The central space in Fig. 1 is key. It is called Disorder, the state of not knowing which system we are in. The danger of being in disorder, is that we tend to interpret and assess the situation according to the system we are most comfortable with, i.e. we compete to interpret this domain according to our preference for action. For instance, those people most comfortable with order seek to enforce rules, and experts seek to conduct research. This can be dangerous. Unfortunately, it is not uncommon to be in the domain of Disorder.

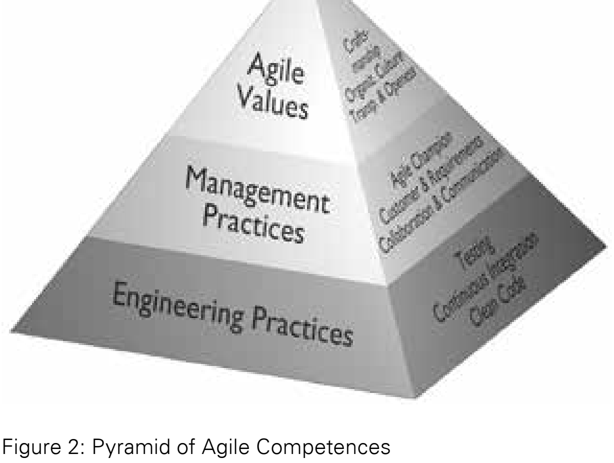

Figure 2: Pyramid of Agile Competences

Agile Competences

Developers in agile software development should possess a set of different competences. These competences can be divided into three categories as shown in the Agile Pyramid of Competences (Fig. 2), which was first presented in [7]:

- Engineering practices
- Management practices
- Agile values

The pyramid initially came from an educational point of view but proved to be useful as a guideline for the kind of competences an agile developer should have, and thus also for the questionnaire and interviews in the study. The analogy of the pyramid visualizes the decreasing number of required skills or competencies for an agile software developer from bottom to top. On the other hand, it reflects the increasing difficulty to learn or acquire these competences. Engineering practices can be taught very well in the classroom or be learned by the individuals on the job at their own pace. Management practices are, in our experience, best learnt through projects in teams. The most difficult competences to teach or learn are the agile values on top of the pyramid, since they often require a change in the attitude of the individual.

With the pyramid we also visualize the order in which the different practice levels should be acquired. Learning the management skills before a developer has a good working knowledge of the underlying engineering skills is like trying to build a pyramid from top to bottom. It might be possible, but at a very high cost (unfortunately, many companies are using this approach: They send their developers to a two day Scrum Master course and expect them to be agile after that). The same is valid for the agile values. Developers usually hear about the values in an Agile or Scrum course. But to properly understand and act accordingly requires a lot of experience and time.

Relationship between CAS and Agile Competencies

We found that the Theory of Complex Adaptive Systems and the Pyramid of Agile Competences are related. The practices at the bottom of the pyramid are typically in the ordered domain, i.e. obvious or complicated (Cynefin framework). There is a lot of expert knowledge necessary to master the engineering practices and that is the reason why it takes a long time for the students or developers to acquire it.

For instance, if the *good practice* of "Automated Unit Testing" is properly applied, there is a relation between cause and effect, e.g. the more (good) tests you write, the higher the quality of the software gets. And the software is also easier to refactor. It is well known that many of the XP-practices like "Automated Unit Testing" and "Refactoring" positively reinforce themselves.

This causality has been well described in [8]. In general, most engineering practices are in the complicated domain where expert knowledge is required and *good practices* can be applied. However, one might argue that some practices are in the simple space. "Clean code" for instance, may be considered an example of a practice in the simple domain.

Moving up the pyramid we get to the management practices. On this level there are more people involved and things start getting more complex. Communication outside of the team within the organization or with customers becomes important. Customer relationships are difficult to manage and so there is a transition between the ordered and the complex domain. There are no simple recipes to follow. Strategies that worked for the last project might be useless or even dangerous for the current project. When the agile (project) leaders are aware of this, they can take appropriate measures and apply *emergent practice* in order to adequately keep the project on track.

On top of the pyramid are the agile values or culture. Cultural aspects are definitely out of the ordered domain and belong to the complex domain. In this domain, the agile leaders and their teams are often learning by doing. Or more precisely, they must dampen the parts that fail and amplify the parts that succeed.

In [7] the authors argued that teaching and learning the agile values on top of the pyramid is much more difficult than the practices at the bottom. With the theoretical insight of the Cynefin framework, it is not difficult to understand why. Moving up the pyramid means transitioning from the ordered to the complex domain (or if you are unlucky: to chaotic domain).

Findings

It should be clear by now, that a list of good or best practices, which the team carefully follows, is not sufficient. It takes more for a project to become a success. In this chapter we present such a list of practices and additionally an overview of similarities or patterns, which we identified in the examined projects. These patterns appear with different characteristics. In some projects the patterns were strongly formed, in others less. These patterns are not meant as a recipe to follow. They are merely a collection of *emergent practices*. As we have seen in complex adaptive systems, we need to create *safe to fail experiments* and act depending of the outcomes. The observed emergent practices may be useful as a starting point to create safe to fail experiments in other projects.

We present the findings following the three levels of the Pyramid of Agile Competencies. We start at the bottom with the engineering practices.

Engineering practices

In the interviews, all the teams reported that they emphasize the engineering practices. These practices are seen as very important and as a kind of foundation, which ensures that software can be developed in short iterations with the required quality. There is a constant process of improvement and refining of these practices. Depending on the maturity of the team and organization, this is triggered by the agile champion (see below), the team members or, in the case of an agile company, even by the management. This can happen ad hoc or formalized. Engineering practices are mostly well-known good or best practices. Many of them have been popularized in eXtreme Programming [8]. A quantitative overview can be found in [1], [3].

Refactoring, automated tests, continuous integration and deployment, pair programming, test-driven development [18], etc. are the tools of the trade [19]. Testing, continuous integration and clean coding was mentioned as core practices in most organizations.

1. *Testing:* In almost all the organizations automated testing on unit level is well established and is seen as an absolute must for providing a good software quality. More mature organizations also apply automated testing on acceptance testing level using new approaches like Automated Acceptance Testing (ATDD) and Behavior Driven Development (BDD).
2. *Continuous Integration:* Continuous integration is seen as an absolute must for being able to deliver software with high frequency. Thus all organizations have established an automated build and test environment which provides immediate feedback to the developers about the quality of the system being built.
3. Some organizations already strive for continuous delivery to automate the delivery process for faster and safer delivery to customers with less effort.
4. *Clean Code:* Continuously paying attention to writing good code from the very beginning is considered more and more important. Some of the organizations started applying the Clean Code [23] approach with the goal of establishing a constantly high code quality. These teams also apply further practices like continuous quality control, regular or even institutionalized code reviews, and regular pair programming.

Management Practices

All teams in our study originally started with Scrum [17]. However, most teams have carefully altered Scrum over time and customized it to the specific needs of their organizations and projects. The ability to alter the process is important for the team for several reasons: First of all it gives the team members the feeling that they are the masters of the process, not vice versa. It objectively allows the elimination of impediments that hinders the team of being efficient. Secondly, it shows that Scrum seems to be a good choice to start with, but that it is not sufficient for all situations and thus should be adapted to the various needs [19]. Retrospective is a good example of how to handle the *dampening* of the parts that fail and the *amplifying* of the parts that succeed.

The interviewed companies especially emphasized the following management aspects:

1. *Customer and Requirements:* In all the interviews, it was pointed out that an intensive and frequent communication with the customer was of outmost importance.
   Agile teams are implicitly or explicitly aware of the fact, that technology sometimes makes solutions possible, which previously nobody (e.g. the users) had seen. Because of this gathering user requirements can be a problem. The users do not know what they want if they do not even know that it exists. It is important to note, that software is developed in a co-evolutionary system with technology.
   In all the successful projects there was a very good understanding of the needed requirements by all team members.
   Communication solely with the Product owner is unsatisfying. Developers feel the need to communicate directly and get feedback from the end-users.
   User Stories [16] is the premier method to express functional and even non-functional requirements on cards.
2. *Agile Champion:* Leaders on all levels of agile organizations need to adopt a Catalyst Leadership style. These leaders thrive by inspiring others without losing the cohesion within the entire system. They know they can trust the organization and its individual members. Most importantly they know, at least by own experience, that software development takes place in several domains at the same time.
   In the interviews we found, that in all the projects there was a least one person that was championing agility and we therefore use the term *Agile Champion* to refer to the role of that person. Why is such a role needed? When an organization wants to become agile, change is inevitable. Of course, it would be nice if positive change happened magically with no effort. Unfortunately, experience shows that it does not. On the contrary, change is very challenging.
   What is the role of the agile champion? Table 4 lists several important activities that require huge amount of talking to people. The role of the agile champion is not to tell them what to do, but to inspire them with a vision of where they and the project could be. It is more a role of pulling than of pushing.
3. *Collaboration and Communication:* Intensive and open communication among all stakeholders is seen to be one of the key elements for successful agile projects. In our interviews we identified three major communication scenarios. The team members themselves must communicate among each other intensively. The team as a whole must communicate with the customer and end-users, and finally the team must establish good communication with the management (which is often organized in a classical hierarchical way).
   To foster communication in teams, almost all organizations implement co-located teams in an open office environment. Most communication is then carried out informally over the desk. This helps to reduce official meeting time significantly and make the necessary meetings much more efficient.
   Regular communication with the customers is established through short iterations with reviews and continuous feedback loops. However, developers have a clear need to communicate directly with end-users instead of the product owners.
   Communication and collaboration with remote teams is always seen as challenging. Companies invest a lot to establish a good communication between these teams. Means are regular chat sessions, remote participation in meetings, and video conferencing. Despite its high costs, some companies value bringing the people together physically at regular intervals. A week of common work usually "refreshes" relationships for about 5-6 weeks, and then the next physical meeting is due.

leading and inspiring agility
helping define which change is necessary
convincing and bringing on board others to support the change
showing that changes are happening and giving good results
avoiding "cowboy" agile and backsliding to the former approach (e.g. waterfall)
leading modifications to the change
removing impediments from the change
leading people to the next level if it starts to plateau

Table 4: The role of an agile champion

Agile Values

All interviewed organizations realized that developing agile is not only a matter of changing processes, but foremost, a change of the culture in an organization. Thus values become more important. Scrum defines the following five Scrum values [17] (in no particular order): *Commitment, Focus, Openness, Respect* and *Courage*. Extreme Programming [8] defines the values a bit differently: *Simplicity, Communication, Feedback, Re-*

spect and *Courage*. In the Agile Manifesto [12] the four core values are: *Individuals and their interactions, delivering working software, customer collaboration* and *responding to change.* Of course there are also other important values like trust, safety, security, quality-of-life etc.

All of these values have in common that they are important agents in complex adaptive systems, i.e. agile software development projects. Because we each interpret the values differently as individuals and as teams, we need to take a look at each value and decide as a team what that value means to us. What is most important is that the team behavior is aligned to the team values. Below we present an overview of the most important findings.

Transparency and Openess

Transparency and openness are highly valued in agile software development. Different measures are taken by the team in order to keep the organization as well as the customer informed about progress and to get feedback quickly. For instance, in Scrum there is the Scrum board and the daily standup meeting, where "the whole world" is invited to attend.

In the study we found that in all projects both aspects were very important. However, transparency can be either good or bad. Sometimes, transparency can prevent people from innovating [9]. Agile leaders know that for news things or innovative ideas to be created, sometimes privacy and protection is needed, because otherwise people outside the team look at them and there is a risk that these ideas get prematurely killed. Therefore, they create protected spaces to allow for new ideas to evolve. So, the right amount of transparency depends on the context of the project, the culture of the company and should be carefully balanced by the agile champion.

Openness of the individuals also means, that the team members not only take responsibility for a dedicated area of the project, e.g. requirements, components, etc., but are open and willed to understand the system as a whole, to have the big picture in mind. People in agile teams take responsibility for what they do in the context of the project as a whole. Agile organizations appreciate that they are (complex) systems made up of self-aware individuals and their members understand, at least implicitly, that each of their interactions may affect this system in potentially unpredictable manners. Again, the theory behind this is well described in complexity theory, i.e. complex adaptive systems. Software development is a co-evolutionary service and not a simple manufacturing process.

Another aspect of openness is the will and capability to learn. As the market place is always changing, agile organizations deliver value through the process of learning. Any change in the organization is based upon continuous learning through successful and failing experiments. In other words change happens as a sequence of learning events that combine to create paramount value, rather than by executing a master plan toward a static goal.

Organizational Culture

It is important how the organization and the company that encompass the agile team and project are organized. Principally there are three possibilities:

1. Agile team, organization and company;
2. Agile team and organization, non-agile company;
3. Agile team, non-agile organization and company.

In the study, we found that seven projects started out with the third option and after some time and effort managed to move to stage 2. In the other project both organization and company were agile. In non-agile companies we observed some tension between the different cultures. Some of the typical problems are differences in hierarchy, responsibility, openness, trust, reporting, management, compensation, planning etc. For instance, if agile values such as transparency and openness lead to promotions and praise in the company, those behaviors will be the ones which individuals select. But if they are at odds with the organizational culture, they will not, and no agile culture can evolve.

Agile project leaders know that they cannot change the company, at least not in the short term. We found that usually they respond by creating an agile environment within the company as well as possible and proactively manage the boundaries. This requires a lot of skills and patience from the agile leader.

Craftsmanship

The idea of highlighting the importance of software development practice has been popularized in [13]. The author introduces the disciplines, techniques, tools, and practices of software craftsmanship. Craftsmanship is much more than a technique: It is about attitude. The author shows how to approach software development with honor, self-respect, and pride; work well and work clean; communicate and estimate faithfully; face difficult decisions with clarity and honesty; and understand that deep knowledge comes with a responsibility to act [14].

In our interviews, we found that members of agile teams and organizations take pride in their work. They seek mastery in their (programming) skills. They know the importance of producing high quality software and great value. In the interviews, all the teams and organizations pointed out that craftsmanship, or whatever they named it, was important and that most developers had a high motivation to become better experts. We found that they challenge themselves and they challenge their colleagues to continuously grow and develop.

Conclusion

In this paper, we have argued that software development is a multi-domain problem, i.e. problems are in the ordered or complex domain. When different people or groups of people are involved, we are typically dealing with a complex (adaptive) system. In such systems, there is no or only weak causality. This means there are no recipes for success. What worked in one project might not work in the next one, or even worse, it might be outright dangerous. When people are not aware of this, they tend to apply best practices or call in an expert. Experts and best practices only work in the ordered domain but not in the complex domain. In the complex domain, a different approach is required.

In the qualitative interviews we saw that successful agile teams and team leaders are actually applying different strategies. They are - implicitly or (seldom) explicitly - aware that software development is a multi-domain problem and act accordingly. In practice, they find that the agile method they are following does not always work and they try to find a different solution strategy. They amplify what works and dampen what does not work.

Agile teams think differently about different problems. There is no one size fits all approach and the action they take depends on which domain the problem is in.

Further work

Our interview study has generated a deep insight about successful agile projects - but also raised new questions.

Does it help the agile team to know more about complexity theory and especially its application in software development? There is the hypothesis that development teams who understand the theory of decision-making frameworks like Cynefin can in practice deliver better software products faster and cheaper. This combination of theory and practice is often referred to as *praxis*. Further research to validate or reject this hypothesis is necessary.

Organizational culture for agile teams, organizations and companies is another huge topic. There are some studies [21], [22] but further work is required. What is the influence of agile organizations in non-agile companies on key aspects like innovation, talent, motivation or customer satisfaction? What happens to innovation? How can agile and non-agile teams and organizations and companies co-exist, does the whole company have to be organized in an agile manner?

References

[1] Version One. State of Agile Development Survey results. http://www.versionone.com/state_of_agile_development_survey/11/, 14.11.2013

[2] S. Kaltenecker et. al. Successful Leadership in the Agile World – a Study Report of PAM. 2011. (German only)

[3] Martin Kropp, Andreas Meier, Swiss Agile Study - Einsatz und Nutzen von Agilen Methoden in der Schweiz. (German) www.swissagilestudy.ch, 20.10.2013

[4] David J. Snowden, Mary E. Boone A Leader's Framework for Decision Making. Harvard Business Review, November 2007, pp. 69–76

[5] D. Snowden, C.F. Kurtz. The new dynamics of strategy: Sense-making in a complex and complicated world, IBM Systems Journal, Volume 42, Number 3, 2003, pp.462-483

[6] R. Arell, J. Coldewey, I. Gat, J. Hesselberg, Characteristics of Agile Organizations, Agile Alliance, 2012, http://www.agilealliance.org

[7] M. Kropp, A. Meier, Teaching Agile Software Development at University Level: Values, Management, and Craftsmanship, CSEE&T 2013, San Francisco

[8] Kent Beck, Extreme Programming Explained: Embrace Change. Addison-Wesley, 2004 ISBN 0-321-27865-8

[9] D. Snowden, Keynote: Making Sense of Complexity, 5. Lean Agile Scrum Conference, LAS 2013, Zurich. http://www.lean-agile-scrum.ch/wp-content/files/2013/Kynote_Making%20Sense%20of%20Complexity%20-%20Dave%20Snowden.pdf

[10] Complex Adaptive Systems http://en.wikipedia.org/wiki/Complex_adaptive_system, Retrieved 18.Aug.2014

[11] http://commons.wikimedia.org/wiki/File:Cynefin_as_of_1st_June_2014.png, Retrieved 18.Aug. 2014

[12] Agile Manifesto. http://agilemanifesto.org/, Retrieved 20.Aug.2014

[13] Robert C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship, 2009, ISBN 0-13-235088-2

[14] Robert C. Martin, The Clean Coder: A Code of Conduct for Professional Programmers, Prentice Hall, 2011, ISBN 0-13-708107-3

[15] Mike Cohn, Agile Estimating and Planning, 2006, ISBN 0-13-147941-5

[16] Mike Cohn, User Stories Applied, For Agile Software Development, 2004, ISBN 0-321-20568-5

[17] Ken Schwaber, Mike Beedle. Agile Software Development with Scrum, 2001, ISBN 0-13-207489-3

[18] Kent Beck, Test-Driven Development: By Example. Addison-Wesley, 2003, ISBN 0-321-14653-0

[19] Henrik Kniberg, Scrum and XP from the Trenches. How we do Scrum. An agile war story, 2007, ISBN: 978-1-4303-2264-1

[20] Alistair Cockburn, Jim Highsmith, Agile Software Development: The People Factor, Computer, vol. 34, no. 11, pp. 131-133, November, 2001

[21] Cristiano Tolfo1, Raul Sidnei Wazlawick, Marcelo Gitirana Gomes Ferreira1, Fernando Antonio Forcellini1, Agile methods and organizational culture: reflections about cultural levels, Journal of Software Maintenance and Evolution: Research and Practice Volume 23, Issue 6, pages 423–441, October 2011

[22] Juhani Iivari, Netta Iivari, The relationship between organizational culture and the deployment of agile methods, Information and Software Technology Volume 53, Issue 5, May 2011, Pages 509–520

[23] Robert C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship, Prentice Hall, 2008