Agile Success Factors

A qualitative study about what makes agile projects successful

Prof. Martin Kropp
University of Applied Sciences and Arts
Northwestern Switzerland

Andreas Meier
Zurich University of Applied Sciences
Switzerland

swiss
agile
study

Authors

Andreas Meier
Zurich University of Applied Sciences (ZHAW)
Winterthur

Prof. Martin Kropp
University of Applied Sciences and Arts
Northwestern Switzerland (FHNW)
Windisch

Publisher: FHNW and ZHAW

Date of publication: May 2015

Website: www.swissagilestudy.ch

ISSN: 2296-2476

1 MANAGEMENT SUMMARY

Various studies show great improvements in software projects when agile software development is applied. However, there are still remaining problems and there are also reports about project failures in the agile community. This raises the question of what factors distinguish successful agile software projects and teams from less successful ones?

The authors of the Swiss Agile Study wanted to shed some light on these questions. We conducted a qualitative interview study with eight successful agile IT companies. We asked them about the essential success factors in their agile projects. The findings are divided into three different categories: Engineering practices, management practices and the values, or culture, they live.

On the engineering level it was found that these companies apply many technical practices in a very disciplined way, with a strong emphasis on quality assuring practices like unit testing, continuous integration and automation, and clean coding.

On the management level it was pointed out that clear requirements, which are verified and validated in very close collaboration with the customer, are essential. The same was true for very close communication within the team. The third aspect that was found, was that in each successful team there was a kind of *Agile Champion* who motivated and inspired the team to use agility.

On the value level we found that successful agile teams live a culture of openness and transparency. They establish an agile culture at least on the team and organizational level (we found only one company who had established the agile method in the whole company). Third, they live an attitude of craftsmanship, being proud of their work and striving for high quality work.

Finally we noticed, that while putting high emphasize on the above practices, mature agile teams start adapting these practices and the agile process to their needs, when they notice that some of the practices do not work or that following the recipe is insufficient. A constant probing, sensing and appropriate responding was observed. This is the typical pattern for moving forward in *complex adaptive systems*. Applying a sense-making methodology like the Cynefin framework, theoretically explains the observations in the present study. Companies should therefore be aware, that software projects are often located in the complex domain, i.e. can be modeled as complex adaptive systems. These kinds of problems rather require *emergent practices* instead of good or best practices and an understanding of the implications of complexity theory is of merit.

2 INTRODUCTION

2.1 Motivation

The Swiss Agile Study, an online survey on the state of software development in Switzerland, was conducted in 2012 and 2014. The studies clearly show significant improvements as compared to traditional approaches, i.e. faster time-to-market, improved change management, and higher satisfaction with the overall process. However, the studies also show, that a significant part of the agile projects fail due to various reasons, though most of the recommended good practices for agile development were applied. So the studies also raised new questions. The predominant question was: What makes agile teams successful? Which factors distinguish successful software projects and teams from unsuccessful ones?

2.2 Scope and Goal of the Study

With the new study the authors wanted to find out, what the essential factors for successful agile projects are. For this they conducted an interview study among eight Swiss IT companies about their most successful agile projects.

The main goals of the study were to find out:

- What the essential criteria are that make an agile software project successful
- If patterns of behavior can be identified for successful projects
- If these patterns can help others conduct successful projects

3 ACKNOWLEDGMENT

The authors would like to thank the Swiss *Hasler Foundation*, www.haslerstiftung.ch, which has generously funded the *Agile Success Factors* project. In 2012, the Hasler Foundation also funded our first *Swiss Agile Study*, a quantitative study on software development in Switzerland, which has run now in a bi-annual survey. The *Swiss Agile Study* was the inspiration for this project.

We would also like to thank all the participating companies of the study for taking the time and their willingness to answer all our questions. Last but not least we would like to thank Jenny C. Ivarsson for proofreading and improving our study.

HASLERSTIFTUNG

4 OVERWIEW OF THE INTERVIEW STUDY

4.1 Interview Method

The aim of our study was to show by example how agility is successfully realized in various kinds of projects and different kinds of companies and branches. To do this 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 considered the described project as successful – from their own point of view and from their customers' point of view.

We conducted eight semi-structured individual interviews and used a self-developed interview guide. We structured our interview questions according to the Agile Competence Pyramid (see Figure 1) with the three different 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. The following table gives an overview of the industry branches covered.

Table 1. Branches covered in study

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

The interviews were conducted with a total of nine participants (at one company, we had two interviewees). The participants were mostly project leaders, group leaders, or department leaders. The interview duration was between one and almost two hours. All interviews were audio recorded and later transcribed. The transcription facilitated the evaluation of the interviews with a statistical text-analysis tool.

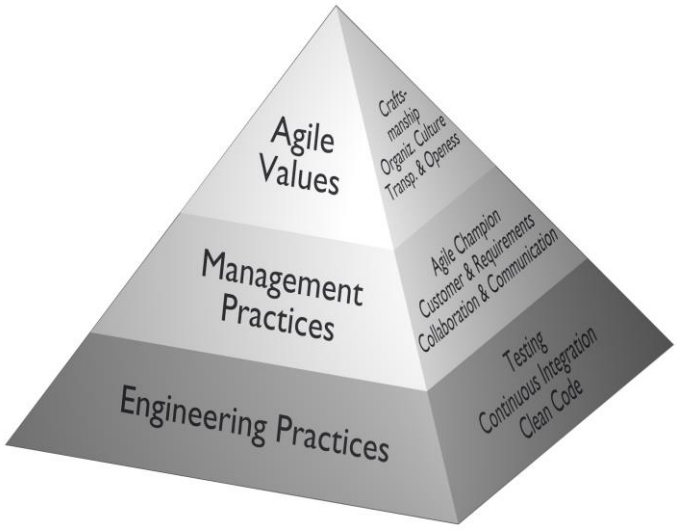

Figure 1. Pyramid of Agile Competences (from [7])

4.2 Organizational Level of Agility

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

Team refers to the software developers, testers, Scrum Masters, Product Owners etc., i.e. the people who 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 in becoming 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 of the companies we questioned 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. This means that those team 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.

4.3 Interview Questions

In preparation for the interview the interviewees were asked to send us a short description of the selected project, including some basic information like duration, effort, and team size. Additionally, they had to answer the following 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 the company and the project (industry sector, company size, in-house/contract work/product development, main project technology).

We asked the questions listed in Table 2 about the topics, Success, Engineering, Organization and Management, Culture and Values, and Improvements.

TABLE 2. INTERVIEW QUESTIONS

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?

5 RESULTS OF THE QUALITATIVE STUDY

5.1 Basic Project Figures

Table 3 shows 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 scrum-of-scrum teams.

5.2 What is Success in Projects?

The question about why the selected project was considered successful was answered in very many different ways.

TABLE 3. PROJECT FIGURES

	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	Manufactur-er	Manufac-turer	IT Service Provider	Insurance	IT Solution	Manufac-turer
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)

The most frequent 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

5.3 Engineering Practices

In the interviews, all teams reported that they emphasize applying 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]. Unit testing, continuous integration and clean coding were mentioned as the core and most important practices in most organizations.

5.3.1 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).

5.3.2 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. Some organizations already strive for continuous delivery to automate the delivery process for faster and safer delivery to customers with less effort.

5.3.3 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 [13] 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.

5.4 Management and Organization Practices for Success

The success criteria on management level that were mentioned most often were the short iterations (typically two weeks) and the self-organization of teams. Other important issues mentioned were User Story Grooming meetings during the Sprint;

close communication with externals; open office. Management support was also argued to be very important by one company. For one company, that has remote teams, daily meetings with visual communication tools like Skype was very important.

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. The interviewed companies especially emphasized the following management aspects:

5.4.1 Customers and Requirements

In all the interviews, it was pointed out that an intensive and frequent communication with the customer was of utmost importance.

Successful agile teams are implicitly or explicitly aware of the fact, that technology sometimes brings solutions previously unknown to the users. Because of this, gathering user requirements can be a problem. The users cannot know they want something 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 with and get feedback from the end-users.

User Stories [16] were the premier method used to express functional and even non-functional requirements on cards in the interviewed companies.

5.4.2 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 (see appendix A) at the same time.

In the interviews we found, that in all the projects there was at least one person who 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?

- To lead and inspire agility
- To help define which change is necessary
- To convince and bring on board others to support the change
- To help show that the change is happening and is bringing good results
- To avoid “cowboy” agility and backsliding to the former approach (e.g. waterfall)
- To lead modifications to the change
- To remove impediments from the change
- To lead the people to the next level, if it starts to plateau

This all requires huge amounts 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.

5.4.3 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 the following three major communication scenarios. Firstly, the team members themselves must communicate intensively with each other. Secondly, the team as a whole must communicate with the customers 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 at the desks. 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 the end-users instead of the product owners.

Communication and collaboration with remote teams is always seen as challenging. Companies invest a lot in establishing 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.

5.5 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, Respect* 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.

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.

5.5.1 Transparency and Openness

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 pre-

vent people from innovating [9]. Agile leaders know that for innovative ideas or things to be created, sometimes privacy and protection is needed, because otherwise people outside the team may challenge the ideas and there is a risk that they get prematurely rejected. 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 and 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 and components, 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.

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.

5.5.2 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 eighth project both organization and company were agile (option 1). 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 the agile values are at odds with the organizational culture, they will not be reinforced, and no agile culture can evolve.

Agile 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.

5.5.3 Craftsmanship

The idea of highlighting the importance of software development practice was popularized by R. Martin in [13]. The author introduces the disciplines, techniques, tools, and practices of software craftsmanship. Craftsmanship is much more than a technique: It is an 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 to increase the business value. In the interviews, all the teams and organizations pointed out that craftsmanship, (or the equivalent thereof), was important and that most developers were highly motivated to become better experts. We found that they challenge themselves and their colleagues to continuously grow and develop.

6 OTHER FINDINGS

6.1 Changing the process

When analyzing the interviews and examining how the teams organized themselves and behaved in their successful projects, we observed that all teams in our study originally started with Scrum [17]. However, most teams have carefully altered their processes and practices over time and customized them to the specific needs of their organizations and projects.

Some of the changes and extensions we noticed, are:

- Introducing User Story grooming
- Changing Sprint planning 1
- Installing Product Owner proxies
- Using Use Cases and User Stories together
- Establishing an Agile Champion
- Bringing distributed teams together
- Introducing testing days

6.2 Reasons for Changes

The ability to alter the process is important for the team for several reasons:

- It objectively allows the elimination of impediments that hinders the team from being efficient.
- Secondly, it becomes apparent that Scrum is a good choice to start with, but that it is not sufficient for all situations and thus should be customized 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.
- Thirdly, and also very important: the team members get the feeling that they are the masters of the process, not vice versa.

6.3 From Best to Emergent Practices

It seems that a list of good or best practices, which the team carefully follows, is not sufficient. It takes more for a project to become successful. Mature agile teams adapt these practices and processes to their needs. We observed a kind of constant probing and sensing and appropriate responses. In other words, the participating organizations do not see these practices as a recipe to follow. They are merely a collection of *emergent* practices (see appendix A). This is the typical pattern for moving forward in *complex adaptive systems* [10] in complexity theory [5]. The teams create *safe to fail experiments* and act according to the outcomes. The observed emergent practices may then be useful as a starting point to create safe to fail experiments in other projects.

Applying a sense-making methodology like the Cynefin framework, explains the observations in the present study theoretically. Therefore, it is suggested that companies be aware, that software projects

are often located in the complex domain, i.e. can be modeled as a complex adaptive system. These kinds of problems rather require *emergent practices* than good or best practices and an understanding of the implications of complexity theory is of merit.

6.4 Theoretical Background

As mentioned in the previous chapter, the observed behavior of successful agile teams is typical for the behavior in *complex systems*. Complexity theory deals with understanding how organizations cope with conditions, constraints and uncertainty and how they adapt their behavior accordingly [23].

The complexity theory of constraints distinguishes between chaotic systems, where no constraints exist, complex systems, and ordered systems, which are highly constraint. Depending on which system your problem is in, the solution is clear and you can either follow simple recipes to solve your problem, or there is no known solution and you have to experiment to a large extent. *Complex Adaptive Systems* are special kinds of complex systems in the sense that they adapt to the changing environment (typical for software projects) [10]. Based on the theory of complex systems and Complex Adaptive Systems, D. Snowden has developed the Cynefin framework [5], which provides some guidelines on how to behave in the various system spaces. A more elaborate discussion of the complexity theory and how it relates to software development can be found in appendix A.

6.5 CAS and Agile Competencies

We found that the Theory of Complex Adaptive Systems and the Pyramid of Agile Competences (Figure 1) are related. The practices at the bottom of the pyramid are typically in the ordered domain, i.e. obvious and complicated (see Cynefin framework in appendix A). A lot of expert knowledge is 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, i.e. the more (good) tests you write, the higher the quality of the software. Additionally, 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, some practices can be placed in the simple domain. An example of this is “Clean code”.

Moving up the pyramid we get to the management practices. On this level there are more people involved and things are more complex. Communication outside 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 in past projects 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 the top of the pyramid are the agile values or culture. Cultural aspects are definitely not in the ordered domain but belong in the complex sphere. In this domain, the agile leaders and their teams often learn 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 the 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 even to the chaotic domain.

7 CONCLUSION

The study shows 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 apply various strategies. They are - implicitly or (seldom) explicitly - aware that software development is a multi-domain problem and act accordingly. In practice, they sometimes find that the agile method they are following does not work and then 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.

8 REFERENCES

[1] Version One. State of Agile Development Survey results. http://www.versionone.com/state_of_agile_development_survey/11/, Retrieved 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/ Keynote_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, ACM Digital Library, 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] http://en.wikipedia.org/wiki/Complexity_theory_and_organizations, retrieved 30. April 2015

9 CONTACT

Andreas Meier

Dozent für Informatik
Zürcher Hochschule für Angewandte Wissenschaften (ZHAW)
Institut für angewandte Informationstechnologie
Steinberggasse 13, CH-8401 Winterthur
E-Mail: andreas.meier@zhaw.ch

Prof. Martin Kropp

Dozent für Informatik
Fachhochschule Nordwestschweiz (FHNW)
Hochschule für Technik – Institut für Mobile und Verteilte Systeme
Steinackerstrasse 5, CH-5210 Windisch
E-Mail: martin.kropp@fhnw.ch

Information

Website: www.swissagilestudy.ch

11 APPENDIX A - THEORY OF COMPLEX ADAPTIVE SYSTEMS AND AGILE COMPETENCES

11.1 Complexity Theory and Software Development

In recent years complexity theory has been widely discussed in the scientific community. In this paper we argue that some parts of software development projects should be treated as complex systems, i.e. the problems are in multiple domains.

11.1.1 Ordered and Un-ordered Systems

There are different definitions of complex systems. In this paper we use the definition of constraints [9]. 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.

11.1.2 Complex Adaptive System

A *Complex Adaptive System* (CAS) is a special case of a complex system. Here the constraints and agents co-evolve. Examples of CAS are the Internet, cyberspace, stock markets, manufacturing businesses, ant colonies, and any human social group-based endeavor. A software development project is therefore 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 a 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 keeping 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 to try 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.

11.2 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 those found 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, 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].

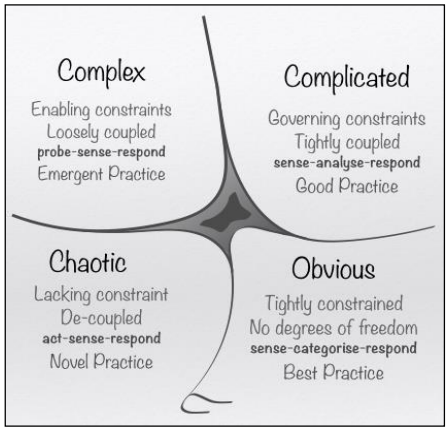

Figure. 2. Cynefin domains (Source [11])

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.

11.3 The five domains of the Cynefin framework

As can be seen in Figure. , 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

[1] There are different definitions of ‘praxis’. Here ‘praxis’ is defined as the combination of theory and practice.

domains (Obvious, Complicated) are those of order, and the left-hand domains (Complex, Chaotic) are those of un-order.

- **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 practice, 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 practice, 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 practice, i.e. new practice and some combination of best and good practice.
- **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 practice*.

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 Figure. 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.

12 APPENDIX B – PROJECT REPORT FORMS

The following tables contain the written project descriptions, which were given by the interviewed companies in advance of the interviews. It especially contains the information about what made the projects successful for the interviewees and what they think made it successful for their customers.

Notes:

1. *For confidentiality reasons the descriptions have been made anonymous and any confidential information has been removed.*
2. *The written descriptions have been translated from German into English*

Project	**P1**		
Company Type	IT Solution Provider	**Project Type**	In-house
Methodology	Scrum	**Project Size [PM]**	24
Duration [M]	3	**Team Size**	5
What makes the project successful from your point of view?			
• We had a dedicated team • The team had the domain knowledge • Highly motivated • Strict timeframe			
What makes the project successful for the customer from your point of view?			
• The labels produced by the software could be directly used in the shops due to the process. • Deadline was reached.			

Project	**P2**		
Company Type	Public service	**Project Type**	In-house
Methodology	Scrum	**Project Size [PM]**	30
Duration [M]	10	**Team Size**	10
What makes the project successful from your point of view?			
• Direct communication among all stakeholder. Especially between the developer and users • Prototype as reference architecture • Common focus on the main goal: First payment on due-date for the whole region was successful • Motto: Keep it strictly simple! • Autonomy and self-responsibility to find solutions and in system design • Very high identification of all stakeholders with the project goals (commitment)			
What makes the project successful for the customer from your point of view?			
• Agreed goals were all reached • Efficient and in-budget			

Project	**P3**		
Company Type	Manufacturer	**Project Type**	Product
Methodology	Scrum	**Project Size [PM]**	30
Duration [M]	12	**Team Size**	5
What makes the project successful from your point of view?			
• Much less hectic development phase • Established refactoring process • No more overtime			
What makes the project successful for the customer from your point of view?			
• Improved robustness • Faster because focused on important features			

Project	P4		
Company Type	Manufacturer	**Project Type**	Product
Methodology	Scrumban	**Project Size [PM]**	100
Duration [M]	9	**Team Size**	3 teams with 4 persons each
What makes the project successful from your point of view?			
• Involve key accounts into prioritization/planning: Ensures that right features are addressed and we understand the requirements. • External tests and early feedback from key accounts: we get better results faster. No late, negative and expensive surprises! • All time requirements and quality requirements were reached. • At least the most important customer requirements were implemented • The ambitious goals spurred the team, motivation and performance was improved			
What makes the project successful for the customer from your point of view?			
• Involve key accounts into prioritization/planning: The product contains the most important requested features. • Frequent delivery of sprint versions: enabled tests and early feedback • Frequent delivery of product increments: generates early value for the customer • External tests and early feedback: Can take influence during the development. In general goals are reached faster, better und more in the sense of the customer. • Trust, that the product will contain the most important feature from the customer's point of view in due time.			

Project	P5		
Company Type	IT Service Provider	**Project Type**	Product
Methodology	Scrum	**Project Size [PM]**	30
Duration [M]	10	**Team Size**	6
What makes the project successful from your point of view?			
• Requirements Engineering • Focus on quality during software development • Very disciplined with respect to all aspects of the project			
What makes the project successful for the customer from your point of view?			
• Requirements Engineering • Transparency • High quality of solution			

Project	P6		
Company Type	Insurance	**Project Type**	In-House
Methodology	Scrum	**Project Size [PM]**	12
Duration [M]	8	**Team Size**	8
What makes the project successful from your point of view?			
• The replacement of the old software system could be done, without any restrictions for the end user. • The teamwork was empowered.			
What makes the project successful for the customer from your point of view?			
• Reduction of the operation costs in the team, and thus • More time for development of new features • Reduction of annual costs of approx. 500‘000 CHF/year and thus higher profit			

Project	P7		
Company Type	IT-Solution	**Project Type**	In-House
Methodology	Scrum	**Project Size [PM]**	72
Duration [M]	24	**Team Size**	5

What makes the project successful from your point of view?

Through the usage of agile artifacts like:

- Backlogs
- Planning in 3 weekly sprints
- Frequent multi team telephone conferences (2x/week)
- 3 weekly sprint plannings (face2face)
- 3 weekly sprint reviews (feedback)

Very good team work (especially for problem solving)

Fast progress of project, progress was visible and measurable

What makes the project successful for the customer from your point of view?

- Customer was continuously involved during development process
- Provided transparency from development side generated trust in the project
- Customer saw results during the development phase and could give feedback

Project	**P8**		
Company Type	Manufacturer	**Project Type**	Product
Methodology	Scrum	**Project Size [PM]**	>120
Duration [M]	ongoing	**Team Size**	5

What makes the project successful from your point of view?

Robustness

- Regression tests are executed for each created baseline
- Each baseline is shipped to the Product Management and a selected group of core users for early feedback and detection of stability issues

User friendliness

- Agile development approach (Scrum) with deliverables for user feedback at an early stage of the project
- User Feedbacks on usability collected via formal feedback channels and informal feedback given by Key Stakeholders

Interoperability

- System Validation

Schedules – Time to Market

- Agile development approach with (approximately) 3 weekly baselines that cover an agreed scope and functionality

Cost-Budget

- Tracking in project development report

Communication

- Proactive communication with Product Management, R&D Stecos

Scope of Delivery

- Early pilot projects defined with the Engineering Center brought proof that the scope of the current release was met

What makes the project successful for the customer from your point of view?

- Reduced project execution costs by substantial engineering efficiency improvements
- Integral engineering process optimization serving the strategy of decentralized project execution