Association for Information Systems
AIS Electronic Library (AISeL)

CONF-IRM 2010 Proceedings

International Conference on Information Resources Management (CONF-IRM)

5-2010

47P. Understanding Agile Software Development in Practice

Karlheinz Kautz
Copenhagen Business School, karlheinz.kautz@rmit.edu.au

Sabine Madsen
Roskilde University, sabinem@ruc.dk

47P. Understanding Agile Software Development in Practice

Karlheinz Kautz
Department of Informatics, Copenhagen Business School, Denmark
Karl.Kautz@cbs.dk

Sabine Madsen
Department of Communication, Business and IT, Roskilde University, Denmark
sabinem@ruc.dk

Abstract

Over the last ten years Agile Software Development (ASD) has received much attention from researchers and practitioners as an approach for dealing with change. However, the proper application area and the use of a mixed - agile and traditional, more plan-driven -approach are still much debated. In this paper, we report from a mission critical project that was considered agile by the involved staff, but which actually employed a mixed agile and plan-driven strategy. We introduce a framework, which allows for (1) descriptive analysis of the project, (2) its discussion against the agile values as presented in the agile manifesto, and (3) a comparison of findings to Complex Adaptive Systems (CAS) theory. We contribute to the debate with rich insight about: which work practices were applied in practice? which of the applied work practices were agile and/or which were more plan-driven in nature? and which of the applied practices fit with CAS theory and/or with a more plan-driven perspective? The analysis of our case shows that some of the agile practices were used in a way that supported both agile values and a traditional focus on processes, documentation, and planning. Moreover, certain traditional practices were in line with CAS theory, while some agile practices fit both CAS and traditional concepts. We suggest that to understand ASD in practice it is relevant to investigate how the applied practices are actually used in the particular case and that the agile manifesto and CAS theory are useful, complementary lenses for doing so.

Keywords

Agile software development, framework, agile manifesto, complex adaptive systems

1. Introduction

Over the course of the last ten years Agile Software Development (ASD) has gained much attention from both the research and practitioner community as an approach that is successful in dealing with change (Dybå & Dingsøyr, 2008). However, ASD is also controversially debated in both literature and practice. The debate concerns, among other things, three interrelated topics, namely:

If and how ASD is different from, the same as, or a compromise between ASD and traditional, more structured development (see e.g. Boehm, 2002; Highsmith, 2000, 2002; Rakitin, 2001; Wang & Vidgen, 2007).

If the agile manifesto and ASD methods are sufficient foundations for studying and supporting practice or if a more theoretical grounding is needed. It has e.g. been suggested that conceptual frameworks as well as general systems and/or Complex Adaptive Systems (CAS) theory are relevant to explore and explain ASD at a higher level (see e.g. Jain & Meso, 2004; Kalermo & Rissanen, 2002; Vidgen & Wang, 2006; Wang & Vidgen, 2007).
If ASD is applicable or not in organizations which are used to the traditional, plan-driven way of working; in distributed development settings; for safety-critical domains; and for otherwise large and complex systems, projects, and teams (see e.g. Avison & Fitzgerald, 2006; Turk et al., 2002).

Recently there has also been a call for more research that looks at how ASD is conducted in practice and how the applicability of agile methods can be extended to cover more broadly (i.e. beyond non-critical software developed by small, co-located teams) (Abrahamsson et al., 2009; Dybå & Dingsøyr, 2008).

In this paper we investigate how ASD is carried out in practice. We introduce and apply a framework to describe and discuss a mission critical project. The project lasted approximately two years and was carried out by a co-located, relatively large team of about 25 members. Both the development and the customer organization considered the development approach to be agile, but in actuality a mixed, agile and traditional, approach was deployed. The framework and the case study are therefore organized around the following research questions: (1) which work practices were applied in practice?, (2) which of the applied work practices were agile and/or plan-driven in nature?, and (3) which of the applied practices fit with a CAS and/or a more plan-driven perspective?

Our contribution to software development research is threefold: firstly, we provide empirical data that indicate that CAS theory is a useful theoretical lens for understanding ASD – as well as other types of software development. Secondly, we show that certain agile and traditional practices created a stable process and product, while other agile practices facilitated quick responses to change and let the product emerge in an only partly predictable way. Thirdly, we suggest that when studying and discussing software development and ASD in particular, it is relevant to look at not only if an applied practice belongs to the agile or the traditional school of thought, but also how it is actually used in a particular case.

The paper is structured as follows. In the next section we present relevant literature and outline the framework that we use to analyze and discuss the case study. Section 3 describes the research approach and section 4 presents an account of the case study that provides contextual information and extracts the applied work practices. In section 5 the applied practices are discussed in terms of the values stated in the agile manifesto. In section 6 the practices are generalized to CAS theory. Section 7 summarizes our answers to the research questions.

2. Background and Framework

The concept of ASD serves as an umbrella for a number of pragmatic approaches, which have emerged based on practice and out of a critique of traditional, plan- and document-driven

development methods (Highsmith, 2002). In the agile manifesto the advocates of these approaches state four values: (1) *individuals and interactions* over processes and tools; (2) *working software* over comprehensive documentation; (3) *customer collaboration* over contract negotiation, and (4) *responding to change* over following a plan (see www.agilemanifesto.org). The agile manifesto recognizes that the concepts mentioned on the right have benefits, but emphasizes the terms to the left. It also provides a set of more operational principles for practical development. Thus, compared to many other systems development approaches, the agile manifesto is a rare example of an approach with stated values and principles (Avison & Fitzgerald, 2006).

Yet, agile methods have been criticised for: (1) representing and promoting an unstructured, undisciplined approach to ISD (Rakitin 2001, Stephens & Rosenberg 2003) and (2) for a lack of grounding in systematic thinking and theory (Conboy & Fitzgerald 2004; Kalermo & Rissanen 2002). Theoretical grounding is necessary because it advances the existing body of knowledge, guides research toward crucial questions, and enlightens practice (Van de Ven, 1989). In line with this, it could be argued that the agile manifesto itself is an attempt to theorize about development as it is actually carried out in practice. Moreover, some ASD proponents (Highsmith, 2002) state that agile approaches draw on Complex Adaptive Systems (CAS) theory as their foundation.

CAS theory has been used to understand and explain agile organizations, processes, and practices from a theoretical and empirical perspective (Jain & Meso, 2004; Vidgen & Wang, 2006). Vidgen & Wang (2006) investigate the CAS literature and identify three principles and six core concepts, which they successfully apply to a case study of an ASD team. Just as the agile manifesto in many ways is formulated to distinguish itself from traditional development approaches Vidgen & Wang's (2006) principles and concepts are explained as a pair of opposites, namely: (1) *time-pacing* (e.g. bi-weekly iterations) vs. occurrence of events, (2) *coevolution* vs. evolution, (3) *working on the edge of chaos* vs. too much or too little structure, (4 and 5) *autonomous agents* working in a *self-organizing*, emergent, and bottom-up manner vs. top-down management, and (6) *working at the edge of time* vs. too much focus on the future (exploration) or the past (exploitation). The agile manifesto and the CAS concepts highlight the need for research and discussions about the extent to which agile development in general and in the particular case is different from, the same as, or a mix of agile and other, more structured, plan-driven development approaches.

When addressing 'the extent to which' it is important to determine the proper object and level of analysis in order to avoid superficial analysis biased in favor of either the agile or the other side of the equation. This is especially important as much of the on-going debate about ASD adopts an either/or perspective (Wang & Vidgen 2007) instead of investigating the phenomenon from different viewpoints and through different theoretical lenses.

In this paper we take 'applied work practices' rather than the information systems development (ISD) organization and process as our object of study. A layered framework (referred to as the ASD framework, see table 1) has been developed to facilitate: (1) analysis of empirical data and presentation of how development was carried out, to identify the used work practices and to understand the multiple and nuanced roles they play, (2) discussion of the applied work practices against the values in the agile manifesto, and (3) generalization of findings to CAS theory.

Object of study	Applied work practices	
Layers	Research question	Perspectives and Key concepts
(1) Presentation and Analysis	Which work practices were applied?	**Structuralist perspective**: description of the structural context, developers, formalized method, and information system **Individualist perspective**: focus on the individual developers' *repertoire*, *language*, and *media preferences* **Interactive process perspective**: interpretation of the *social context*, *social process*, and *content of change*
(2) Discussion against values	Which of the applied practices were agile and/or traditional in nature?	Individuals and interactions over processes and tools *Working software* over comprehensive documentation *Customer collaboration* over contract negotiation *Responding to change* over following a plan
(3) Generalization to theory	Which of the applied practices fit with a CAS and/or a more traditional perspective?	*Time-pacing* vs. events *Coevolution* vs. evolution *The edge of chaos* vs. too much/little structure *Autonomous, self-organizing* agents vs. top-down management *The edge of time* vs. too much exploration/exploitation

***Table 1**: The ASD Framework*

The first layer draws on an existing framework developed and applied to understand and compare the emerging methods in two development projects (Kautz, 2004; Madsen et al. 2006; Madsen & Kautz, 2009). This framework is based on a synthesis of prominent IS literature (Fitzgerald et al., 2002; Schön, 1983; Slappendel, 1996; Walsham, 1993). It consists of three perspectives: (1) the structuralist, (2) the individualist and (3) the interactive process perspective. The structuralist perspective addresses descriptive characteristics of the information system (IS) under development, the formalized method, the involved development team and its members, as well as the project's context. The individualist perspective focuses on the involved individuals' repertoire, language, and preferred media. The interactive process perspective centers around the social context, which addresses social relations, social infrastructure, and social history; the social process, which emphasizes aspects of culture and politics; and the content of the change that takes place in the development project under investigation. The framework has been chosen because its different perspectives and key concepts allow for a movement from surface description of technical, methodical, and organizational characteristics to in-depth interpretation of the social setting, and the multiple and nuanced roles that different work practices play.

The next two layers in the ASD framework take the identified work practices and discuss them against the four agile and traditional values in the agile manifesto and against CAS and traditional concepts.

The ASD framework provides a structure for analyzing and presenting the use of ASD in practice through different perspectives and for discussing the findings from different angles to

arrive at a rich understanding of ASD as an empirical phenomenon. Below, we introduce the research approach and apply the ASD framework to a concrete case.

3. Research Approach

The presented research is based on a qualitative case study (Creswell, 2003) of an ASD project. Our research approach is inspired by Eisenhardt (1989) and Walsham (1995), who stress that in all types of research, including case study research, theory is important as an initial guide to data collection, during the iterative process of data analysis, and as a final product of the research. It is often stated that it is not possible to generalize and certainly not to theorize from a single case study. However, Walsham (1995) suggests that it *is* possible to generalize case study *findings* to theoretical propositions. He outlines four types of possible case based generalizations, namely generation of theory, development of concepts, the drawing of specific implications, and contribution of rich insight. So inspired, we have used the ASD framework to guide our data collection, analysis, and case study presentation in order to contribute to the existing body of knowledge with rich insight about ASD in practice. The empirical data for the case study was collected in semi-structured, open-ended interviews conducted by a team of two researchers over a three day period. The research team performed 12 interviews with 11 individuals – one project manager was interviewed twice. This included nearly a third of the development team and a representative sample of key players and future users from the customer organization. The interviews were tape-recorded and transcribed. For the qualitative data analysis a software tool (NVIVO7) was used. The interview data was supplemented with company and project documents such as method, requirements and release descriptions, and project plans. Data collection, coding, and initial analysis were guided by the key concepts from level 1 in the ASD framework, and the findings from this analysis have been generalized to and discussed in accordance with the theoretical propositions at level 2 and 3.

The ASD project was organized in two phases. When this study was performed phase one had been successfully closed and phase two had been going on for 4 months. Responding to an inquiry call during our data analysis the responsible project manager stated that phase two ended 10 months after our study, on time and budget with all parts of the IS being operational.

4. An Account of ASD in Practice

The project under investigation was concerned with the development of an operations management system (OMS) for the WaterWorks of a large German city by a small software company called AgDev. The system was developed with a web-based graphical user interface and a backend to interface the technical infrastructure as defined by an underlying Enterprise Ressource Planning (ERP) system. The project was organized in 4 subprojects to provide IT support ranging from customer management to the maintenance of the sewer system. The OMS project was described by both the customer and the development company as a success. However, various comments were made about finding the right balance between agility and stability to achieve a viable ASD approach. Below, we present the project in light of the structuralist, individualist, and interactive process perspectives.

The Structuralist Perspective

At the time of the project AgDev consisted of about 25 employees, including 20 developers. AgDev based its development approach on the agile method eXtreme Programming (XP) (Beck & Andreas, 2004). The formalized method included planning techniques for releases

and iterations called planning games, user stories, and story cards to specify user requirements, onsite customers to support customer-developer communication, daily meetings (stand-up meetings) for all project teams, pair programming, re-factoring, collective ownership, and continuous integration and testing to develop the software proper. Software releases were provided every 3- 6 months with each release being organized in iterations of 3 – 6 weeks duration. Working software was also presented to the customer in shorter, weekly cycles to get fast feedback.

AgDev had extended the method with some project management processes to cater for larger projects such as an elaborate overall project plan, formal reporting mechanisms and a formal contract based on a requirements specification produced by the customer.

The OMS project was organized in two phases. In a first 12 months exploration phase prototypes were developed to catch requirements and possible solutions. This led to the development of a comprehensive requirements document by the customer organization and their decision to contract AgDev also for the actual development of the OMS.

In this main development phase a team of about 12 development staff with multiple roles such as project manager, analyst, customer contact, and developer worked onsite in a building owned by WaterWorks. The project team also consisted of a varying number of users with at least one user representing each of the subprojects.

In addition to two project managers, each representing one of the companies, a sophisticated management structure was established. It consisted of one subproject manager, also acting as contact person, from AgDev and one subproject manager, also acting as onsite-customer, from WaterWorks for each individual subproject.

The development team consisted largely of highly educated and motivated, young staff, but only the project managers had extensive experience with agile methods and none of them had ever participated in such a large project.

Individualist Perspective

Although some prior experience with agile methods existed, no team member had worked in an ASD project of this scale and size before. In this situation the experienced developers' repertoire of development techniques as provided by XP and their prior work with more traditional documents as media shaped the development process.

Quite a number of different documents existed, but they were all relatively short and concise. From a customer perspective they were described as follows: *"Well, we have the overall realization concept [the internal term for the requirements specification] as the basis for the contract and as a refinement hereof the requirements lists. These lists govern what should be the outcome of an iteration. … And on the level below there are the story cards, these, so to speak, represent the detailed specifications and plans for the developer's process."* The developers shared this perception and confirmed that the documents, both in length and in number, were adequate.

The predominant media used in the development process were the story cards. They were based on the overall realization concept, which was produced by the customer as a basis for the contract and the requirements lists. The requirements lists were primarily produced by the AgDev project manager and some of the subproject managers, who also worked as contact

persons for their counterparts at WaterWorks and as developers. In the planning game the customers used the requirements lists to describe their requirements for a particular iteration, then the developers created and estimated the story cards in team sessions, and lastly the developers and customers prioritized the cards together. In general, each iteration covered 10 – 12 stories.

Despite the number of documents, the developers used little written material when implementing the story cards. Direct communication between the developers and WaterWorks was preferred and perceived as sufficient. The level of communication and the way feedback was gathered from WaterWorks varied across the four teams. A WaterWorks subproject manager estimated "*that a story card explains 60% of a requirement and the other 40% of it need further enquiries"*, and his AgDev counterpart explained that "*Certain users insist on being contacted by phone while others prefer to be contacted per e-mail.*"

Interactive Process Perspective

The interaction within the AgDev development teams was characterized by trust, close collaboration, and mutual learning. An AgDev subproject manager explained that if "*[i]n a stand-up meeting a developer had a good idea I assigned the task to him. The project manager allows such decisions to be made by ourselves*." The pair programming teams emerged through spontaneous formations, as the same subproject manager put it "*first I look who is free and then I go and work with him*". When implementing the story cards, the developers had much discretion with regard design decisions, but they maintained their relationships and coordinated through daily stand-up meetings where all developers came together to report briefly about their activities. Pair programming and stand-up meetings were also the means whereby developers learned from each other.

The interaction between AgDev and WaterWorks was characterized by social relations that had evolved over time from non-existing to a partnership based on trust. It started in the tendering process, which WaterWorks opened after several attempts of traditional ISD based on a standard ERP system had not led to the desired results.

The social infrastructure beyond AgDev's internal practices within the developer teams - such as daily stand-up meetings and pair programming - can be portrayed as a close collaboration between AgDev and WaterWorks. There were onsite users available, as one WaterWorks subproject manager said: "*My management put me here 100%.*". Further, to achieve a closer relationship between the teams of AgDev and WaterWorks the pairing of the subproject managers was careful thought through by the two overall project managers. It was not by chance that the female AgDev subproject manager's counterpart was a female subproject manager.

The politics of the project were characterized by a relative flat hierarchy and autonomy in the development team. Also, a project champion was appointed, who strongly lobbied the project and the chosen agile approach. But at WaterWorks two further power players existed, which influenced the project: the staff council and the internal IT department. The staff council was perceived as a strong political player because they "*can stop the entire project*" as a WaterWorks subproject manager put it. Involving them in software testing and the presentations of the working software calmed them down. In addition, some employees of WaterWorks' IT department preferred a particular ERP system, which they had been involved to develop, and pressed the case for this system. However these issues did not

dominate and were dealt with by applying techniques, which showed the benefits of the approach and the system under development. Handling change was one such technique.

Change in the ASD project, especially change of requirements was an accepted fact of life. Many change requests were detected through the scheduled acceptance test sessions for an iteration with customer representatives onsite and were then dealt with in the next iteration. Changes also emerged through the weekly and bi-weekly feedback sessions built into an iteration. The close collaboration also had an impact on the culture of the project. The developers, despite their varying experience, made up a quite homogeneous group, which in general was sympathetic to all customer related issues. The frequent feedback loops had the effect that minor misunderstandings were caught and dealt with as changes early before they could grow into something bigger. In each cycle the focus was on the current iteration and on the current user stories while taking the existing working software and future extensions into account.

5. Discussion against Agile Values

The case study account shows that the following work practices were used in the OMS project: short releases and iterations, stand-up meetings, pair programming, planning games based on user stories and story cards, collective ownership, test sessions after and feedback sessions during iterations, onsite customer representatives and other users, a top-down planned and implemented project organization, a two-phased development process, and a formal contract based on a requirement specification.

We now proceed to answer the question of which of the practices were agile and/or more traditional in nature. We consider individuals and interaction, working software, customer collaboration and responding to change as agile values, while we see process and tools, documentation, contract negotiations and following a plan as traditional values.

Individuals and Interaction over Processes and Tools. XP provides a number of processes and tools such as short releases and iterations, planning games, user stories, story cards, onsite customers, pair programming, collective ownership, and stand-up meetings to structure ASD. Planning, releases, and iterations are also part of more traditional development approaches. The OMS project had a top-down planned and implemented project organization, which allowed for interaction and easy access to customer representatives. In the development teams, pair programming was an important process that supported the interaction of the individual developers, but it was not easy to find the right time to change partner and keyboard. Several practices (such as staying with the same partner until a card was implemented, switching keyboard every 20 min., and subproject manager intervention) were tried before the teams found their own rhythm, which did not follow any explicit rule. Stand-up meetings with all teams took place every day. These sessions were originally detailed and long, but this practice was later refined and the shorter meetings were acknowledged as very helpful. Intensive developer interaction took place in the beginning of each iteration, where all story cards were developed and estimated. The mechanisms of pair programming, stand-up meetings, and story card development could not totally provide the intended full collective ownership which the project leader explained with the size of the teams, but they kept the development teams sufficiently informed. We conclude that the applied agile practices can be seen as processes and tools that support development, and are in line with traditional values, while the top-down planned and implemented project organization and the applied agile practices supported a high degree of interaction with users as well as within and between the development teams.

Working Software over Comprehensive Documentation. In the project, working software was the measure of progress and the short releases, iterations, and feedback cycles provided the structure for the development hereof. However, documents were also used. The project was performed based on an elaborate project plan and an overarching requirements specification. Moreover, for each iteration a number of different, but short and concise documents, such as requirement lists and story cards, were outlined. The focus on working software and documents illustrates that both agile and traditional values were in play.

Customer Collaboration over Contract Negotiation. Customer collaboration took the form of onsite development, easily accessible onsite customers and users, as well as telephone contact and email correspondence. The planning games, the presentations of working software, and the acceptance tests were also crucial elements. The overall development process was a two-phased approach structured around a requirements specification, and a development phase. Between these two phases a new formal contract was negotiated. The planning games were partly based on the requirements specification produced by the customer. AgDev did most of the work in the early stages of the planning games as they developed the requirements lists and the story cards, but the level of customer collaboration increased during prioritization and implementation of the story cards. We conclude that the agile practices of onsite development, onsite customers, prioritization and implementation of story cards, presentations of working software, and acceptance tests facilitated customer collaboration, while the more traditional two-phased development approach acknowledged and supported the – for the customer – important role of contract negotiation.

Responding to Change over Following a Plan. Dealing with change was an integral part of the daily activities. The acceptance tests were a major source of change requests; customer representatives regularly performed 'road shows' in the user departments to collect feedback and ideas for improvements; and change requests were brought forward on a shorter time scale via weekly and bi-weekly feedback sessions. These different feedback mechanisms provided structures for collecting ideas and change requests, which were then implemented. There was much focus on responding to change. However, plans and planning also played an important role. Even the weekly feedback sessions were to some extent planned, as were the acceptance tests. The project also used more traditional planning techniques and had an overall long term project plan that covered the whole time period. A more fine-grained plan was developed for the individual iterations, which made up a release detailed to single weeks. The planning game and the story cards then offered the devices for planning at an even more detailed level for very short periods of time. In summary, the agile practices of frequent, continuous acceptance tests, customer representative 'road shows', feedback sessions, and quick implementation of change requests constituted the means for dealing with changes to the *product*. At the same time, the frequent planning sessions structured around releases, iterations, planning games, and story cards performed within the frame of a larger project plan allowed for a well-planned and structured *process*.

In summary, many of the agile practices were applied in a way that supported both agile and traditional values. Moreover, the top-down planned and implemented project organization facilitated a high degree of interaction and collaboration with the customer, while the traditional practices of a two-phased development approach, a formal contract based on a requirements specification, and an overall project plan primarily were in line with traditional values.

6. Generalization to CAS Theory

In this section, we answer the question concerning which of the applied agile and traditional, plan-driven practices fit with CAS and/or with concepts. We consider time-pacing, coevolution, the edge of chaos, self-organizing agents, and the edge of time CAS concepts, while we refer to events, evolution, too much or too little structure, top-down management, and too much exploration or exploitation as traditional terms.

Time-pacing vs. Events. In the OMS project, the use of releases every 3-6 months, iterations of 3-6 weeks duration, formal and informal weekly and bi-weekly feedback sessions, and daily stand-up meetings fit with the concept of time-pacing as they were performed continuously in accordance with an internally set pace. In contrast, the two-phased development approach and the overall project plan were developed at the beginning of the project; project planning and planning games were carried out at the start of each iteration; and test sessions were performed at the end of each iteration and release. We consider these practices event-driven. It could be argued that as the duration and closing dates for the two project phases were pre-planned, the two-phased development approach was in line with the concept of time-pacing. However, these durations and deadlines were not recurring and they were not set internally, but by the customer. We conclude that the agile practices of releases, iterations, feedback sessions, and stand-up meetings are in line with CAS theory, while the traditional two-phased development approach and overall project plan as well as the planning games, and test sessions are event-driven and therefore traditional in nature.

Coevolution vs. Evolution. In the project, the applied agile practices of easy access to customer representatives in the form of subproject managers and other onsite users, onsite development, interaction with customers during prioritization and implementation of story cards, presentation of working software, and test sessions all facilitated the coevolution of all system parts. These practices are also listed as supporting the agile values of customer collaboration. Moreover, the traditional, customer-produced requirements specification supported coevolution as it was read and used by AgDev personnel to outline requirements lists. In contrast, the developer-produced requirements lists and story cards, although based on input from the customer, are more in line with the concept of evolution as the customer representatives and users were not directly involved in these steps of the planning game. However, over time, and especially during coding of the working software the story cards and the details hereof were discussed with the onsite customers. We conclude that in the OMS project both agile and traditional practices supported a high level of interaction between the involved staff from the development and customer organizations, which in turn means that the IS under development can be seen as largely co-created.

The edge of chaos vs. too much/little structure. In the OMS project, short releases and iterations, frequent planning sessions, planning games, pair programming, daily stand-up meetings, test and feedback sessions, helped create a stable process because they were performed continuously in accordance with internally set time intervals or the occurrence of events. These practices structured the developers' day-to-day activities and helped them know what to do, when, and what to expect from others. The same goes for the applied, more traditional practices of a top-down planned and implemented project organization, a two-phased development approach, overall project plan, formal contract, and requirements specification. These practices acted as structuring mechanisms that created a relative stable space within which the development process and the working software could emerge. In contrast, volatility and flexibility were brought about by the following practices that all concern changes to the working software, namely the test and feedback sessions, the other

presentations of working software, as well as the quick implementation of design ideas and change requests. Thus, certain both agile and traditional practices contributed to the creation of a stable process and product, while other agile practices that were applied to respond to change let the working software emerge in an only partly predictable way.

Autonomous, self-organizing agents vs. Top-down management. The development team members acted as autonomous, self-organizing agents when they refined the practices of pair programming and stand-up meetings. In contrast, the project organization and overall project plan were outlined and implemented by the two project managers. In addition, the customer-requested two-phased development approach, the new formal contract negotiated between the requirements and the development phase, and the customer-produced requirements specification can be seen as part of a top-down management approach and therefore in line with the traditional perspective.

The edge of time vs. too much exploration/exploitation. The daily stand-up meetings served the purpose of keeping a focus on today as did the frequent presentations of working software, while the overall project plan, the frequent planning sessions structured around releases, iterations, planning games, etc., and the implementation of working software supported a focus on and constituted a manifestation of both the past and the future. Moreover, the work with the requirements specification, requirements lists, and development of story cards can be seen as an exploration of the future work situation and the future software. It seems however that none of these practices were applied deliberately to be on the edge of time. Instead, in this case, they were used to create a stable process and product and to keep each other in the development and in the customer organization informed. Thus, it can be argued that the applied agile and traditional practices were more in line with a focus on exploitation and therefore the traditional perspective. But, while some exploration and exploitation has taken place, we cannot determine if there was too much exploitation.

The above discussion points out that in particular the concepts of time-pacing, coevolution, the edge of chaos, and top-down management and event, are useful for understanding the OMS project profile and the practices applied.

With the help of both agile and traditional practices the project was organized such that a stable process was created where actually little chaos prevailed and where certain agile practices allowed for a focus on changes to the product. Thus it might be better to follow McKevey (2003) who to emphasis both the importance of structure and stability and of flexibility and chaos, proposes to instead of thinking of the edge of chaos, use the concept of region of emergent complexity, which he calls the region between stability and chaos, when analyzing complex adaptive systems in an organizational context.

Our analysis shows that both the applied agile and traditional practices can be placed in accordance with CAS and traditional concepts. This suggests that CAS theory in general, as well as a comparison of work practices against CAS and traditional concepts as illustrated above are a useful theoretical lens for understanding not just ASD, but also other types of software development.

7. Conclusion

In this paper we report from an empirical case study of a systems development project, the OMS project. The case study account shows that in the OMS project *both* agile practices and

more traditional practices were used. The important agile practices were: short releases and iterations, stand-up meetings, pair programming, planning games based on user stories and story cards, test sessions after and feedback sessions during iterations, and onsite customer representatives and easy access to other users. Important traditional practices were: a top-down planned and implemented project organization, and a phased development process structured around a formal contract and requirements specification.

The research contributes to the on-going debate about ASD with a case study that supports the view that in practice what practitioners consider as ASD is undertaken as a compromise between a purely agile and a more traditional approach. In line with this, the case study account and the discussion of the applied practices against the four values in the agile manifesto also show that some of the applied agile practices (e.g. short releases and iterations and planning games) support or were used in a way that supports both the agile side as well as the traditional accent on processes, documentation, and planning.

The comparison of the applied practices to CAS concepts further demonstrate that in this case the development was performed as a quite planned and structured endeavor. Key findings are that *certain agile and traditional practices (e.g. the releases and iterations with pre-planned deadlines and the requirements specification, respectively) helped create a stable process and product, while other agile practices facilitated quick responses to change and let the product emerge in an only partly predictable way.* The discussion also reveals that some traditional practices (such as the phased development process and the project organization) can be seen to be in line with CAS theory, while some of the agile practices (such as the story cards) supported both CAS and traditional concepts. We conclude that when studying and discussing ASD it is relevant to look at not only if an applied practice belongs to the agile or traditional 'school', but also how it is actually used in the particular case.

The findings point to a number of areas for future research. First, it is relevant to ask which practices are or could be used to support both agile and traditional values as well as CAS and traditional concepts. This further suggests that a mapping of empirically derived and literature prescribed practices against values and concepts is relevant. Such a mapping would provide researchers and practitioners with overview and insights, which in turn might assist researchers in identifying new research topics and suitable theories and practitioners in reflecting on their practice in general and in identifying specific practices relevant for the particular situation in which they find themselves. Second, the current assumption seems to be that CAS theory is a useful theoretical lens for understanding ASD, but not other types of development. The research presented here has led us to wonder if that is the case. We therefore suggest that more research is needed to investigate if CAS theory in general, or certain CAS concepts, also are relevant as a theoretical foundation for understanding other types of ISD, and if so, which types of ISD? and which types of understandings can be achieved?

References

Abrahamsson, P., Conboy, K., & Xiaofeng, W. (2009). 'Lots Done, More to Do': The Current State of Agile Systems Development Research. *European Journal of Information Systems*, 18(4), 281-284.

Avison, D.E., Fitzgerald, G. *Information Systems Development: Methodologies, Techniques and Tools*, 4th ed., McGraw-Hill, Maidenhead, UK, 2006.

Beck, K., Andreas, C. *Extreme Programming Explained: Embrace Change*, 2nd Edition, Addison Wesley Professional, Mass., USA, 2004.

Boehm, B. Get Ready for Agile Methods, With Care. *Computer*, (35:1), 2002, pp. 64-69.

Cockburn, A. *Agile Software Development*. Addison-Wesley, Boston (Mass.), 2002.

Conboy, K., Fitzgerald, B. Toward a Conceptual Framework of Agile Methods. In *Proceedings of Extreme Programming and Agile Methods - XP/ Agile Universe,* Springer-Verlag Berlin, 2004.

Creswell, J.W. Research design - Qualitative, Quantitative and Mixed Methods Approaches. Sage Publications, Thousand Oak, CA, USA, 2003.

Dybå, T., & Dingsøyr, T. (2008). Empirical Studies of Agile Software Development: A Systematic Review. *Information & Software Technology*. 50(9-10), 833-859.

Eisenhardt, K.M. Building Theories from Case Study Research. *Academy of Management Review* (14:4), 1989, pp. 532-550.

Fitzgerald, B., Russo, N. L., Stolterman, E. *Information Systems Development, Methods in Action*. McGraw-Hill, London, UK, 2002.

Highsmith, J. Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. Dorset House Publishing, New York, 2000.

Highsmith, J. *Agile Software Development Ecosystems*. Addison-Wesley, Boston (Mass.), 2002.

Jain, R., Meso, P. Theory of Complex Adaptive Systems and Agile Software Development. In *Proceedings of the Tenth Americas Conference on Information Systems*, New York, August 2004, pp. 1661-1668.

Kalermo, J., Rissanen, J. *Agile Software Development in Theory and Practice*. M.Sc. Thesis on Information Systems Science. University of Jyväskylä, Finland, 2002.

Kautz, K. The Enactment of Methodology: The case of developing a multimedia Information system. In *Proceedings of the International Conference on Information Systems*, ICIS 2004, December 12-15, 2004, Washington, DC, USA, pp. 671-684.

Madsen S. & Kautz K. (2009), A Framework for Identifying the Drivers of ISD Method Emergence, in: *Systems Analysis and Design: Techniques, Methodologies, Approaches, and Architecture.* In Advances in Management Information Systems (AMIS) Monograph Series, (eds.) Roger Chiang, Keng Siau, Bill C. Hardgrave, Chapter 5, pp. 58-71.

Madsen, S., Kautz, K., Vidgen, R. A Framework for Understanding how a Unique and Local IS Development Method emerges in Practice. *European Journal of Information Systems* (15:2), 2006, pp. 225-238.

McKelvey, B. MicroStrategy from MacroLeadership: Distributed Intelligence via New Science. In *Mobilizing the Self-Renewing Organization*, Lewin, A. Y. and Volberda, H. (eds.). Armonk, NY, M.E. Sharp, 2003.

Rakitin, S. Manifesto Elicits Cynicism. *IEEE Computer*, (34:12), 2001, pp. 4.

Schön, D. A. The Reflective Practitioner - How Professionals Think in Action. Basic Books, USA, 1983.

Slappendel, C. Perspectives on Innovation in Organizations. *Organization Studies* (17:1), 1996, pp. 107-129.

Stephens, M. & Rosenberg, D. *Extreme Programming Refactored: The Case Against XP*. Apress, New York, 2003.

Turk, D., France, R., Rumpe, B. Limitations of Agile Software Processes. In *Proceedings of Third International Conference on eXtreme Programming and Agile Processes in Software Engineering*, Alghero, Sardinia, Italy, 2002.

Van de Ven, A. Nothing is Quite So Practical as a Good Theory. *Academy of Management Review*, Vol. 14, No. 4, 1989, pp. 486-489.

Vidgen, R., Wang, X. Organizing for Agility: a Complex Adaptive Systems Perspective on Agile Software Development Process. In *Proceedings of the Fourteenth European Conference on Information Systems*, Ljunberg J., and Andersson, M. (eds.), Goeteborg, Sweden, 2006, pp. 1316-1327.

Walsham, G. *Interpreting Information Systems in Organizations*. Wiley Series on Information Systems, Chichester, UK, 1993.

Walsham, G. Interpretive Case Studies in IS Research: Nature and Method. *European Journal of Information Systems* 4, 1995, pp. 74-81.

Wang, X., Vidgen, R. Order and Chaos in Software Development: A comparison of two software development teams in a major IT company. In *Proceedings of the Sixteenth European Conference on Information Systems*, Winter, R., et al. (eds.), St. Gallen, Switzerland , 2007, pp. 807-818.