BEYOND SIMPLE CLASSIFICATIONS: CONTEMPORARY INFORMATION SYSTEMS DEVELOPMENT PROJECTS AS COMPLEX ADAPTIVE SYSTEMS

Completed Research Paper

Karlheinz Kautz
University of Wollongong, School of Information Systems & Technology
Wollongong NSW 2522, Australia
&
Copenhagen Business School, Department of Operations Management,
Solbjergplads 3, DK-2000 Frederiksberg, Denmark
Karl.Kautz@cbs.dk

Abstract

Contemporary Information Systems Development (ISD) takes place in a dynamic environment and is generally acknowledged as a complex activity. We investigate whether complex adaptive systems (CAS) theory is relevant as a theoretical foundation for understanding ISD, and if so, what kind of conception can be achieved by utilizing the theory? We introduce key CAS concepts and describe an emergent method framework for understanding ISD. Extending existing research, our main contribution is twofold: We first show how CAS and CAS principles are advantageous for comprehending and organizing ISD in general, beyond any particular development approach chosen for the execution of a project such as agile development. Thereby, we contribute to a complexity theory of ISD. Second, we back up our argument with a coherent empirical account of contemporary ISD, and thus contribute with practical advice for ISD derived from this perspective to successfully cope with complexity in ISD in an adaptive manner.

Keywords: Information Systems Development, Complex Adaptive Systems Theory

Introduction

Contemporary Information Systems Development (ISD) is generally acknowledged to be a complex activity (Truex et al. 1999, Highsmith 2000, Meso & Jain 2006). Benbya and McKelvey (2006) present a commendable analysis of sources of complexity in contemporary ISD, and recognize as such: changing user requirements; changing organizational needs; changing external competitive conditions; increased interdependencies among the involved individuals, organizations and technologies; and the rapid evolution of IS and IT. The mentioned authors purport that ISD is often viewed as a complex top-down process and lament the fact that such a perspective falls short in dealing with the identified, but often unexpected contingencies of, the ISD process. As an alternative, they put forward a conception of ISD based on complexity science and theory (Kauffman 1993, Kauffman 1995, Holland 1995, Holland 1998), proposing that ISD projects, following a special branch of complexity science, should be viewed as complex adaptive systems (CAS) and that these projects are better understood through the application of CAS. According to complexity science, complexity generally refers to an emergent property of systems made of large numbers of self-organizing agents that interact in a dynamic and non-linear fashion and that share a path-dependent history (Jacucci et al. 2006 citing Cilliers 1998). CAS theory is a branch of complexity science that studies how a complex system can be adaptive to its environment and how innovative properties of a system emerge from the interactions of its components (Vidgen & Wang 2009). CAS theory has been extensively applied in management and organization studies, with Benbya and McKelvey (2006) and Vidgen and Wang (2009) providing two excellent summaries of this literature. More recently, there are also general writings on the topic in the IS domain (Desai 2005, Jacucci et al. 2006, Merali & McKelvey 2006), as well as specific research on CAS and ISD (Highsmith 2000, Jain & Meso 2004, Augustine et al. 2005, Benbya & McKelvey 2006, Meso & Jain 2006, Vidgen & Wang 2006, Wang & Vidgen 2007, Kautz & Zumpe 2008, Vidgen & Wang 2009, Wang & Conboy 2009, Kautz & Madsen 2010). In the discipline of ISD, the work has either been conceptual or in the area of agile software development. Vidgen and Wang (2009) relate the scarcity of empirical research on CAS in the IS field to the difficulty of turning the quite abstract concepts of CAS theory into sufficiently concrete case study research. While the authors provide such a study and a CAS-based framework to investigate and improve the organization of agile software development, they simultaneously criticize the often found simple classifications of agile versus nonagile development practices, and call for research that on a CAS theory background, examines closely how development practices are actually implemented beyond such classifications. Kautz and Madsen (2010) provide a similar argument, basing their research on the objective of understanding agile software development. The authors suggest that more research is needed to investigate if CAS theory in general or certain CAS concepts in particular are relevant as a theoretical foundation for understanding ISD, and if so, which kind of understanding can be achieved? To answer these questions, the research presented here follows Benbya and McKelvey (2006); it considers ISD projects as CAS independent of a particular methodological approach chosen for the execution of the project, thereby demonstrating the advantages of such an approach through an empirical case study presentation and analysis. As such, it contributes to a complexity theory of ISD, and provides practical advice derived from this perspective.

The remainder of the paper is structured as follows. The next section presents relevant CAS concepts and outlines the framework that we use to analyze and discuss the case study. Section 3 describes the research approach and the following section provides the case setting as well as an account of the case study that gives contextual information and identifies the applied work practices. In section 5, the findings section, these practices are revisited and linked to CAS theory. In the penultimate section we discuss our results with regard to other work which has recognized CAS theory as a valuable approach to understanding contemporary ISD. In section 7 we conclude by bringing to the fore and summarizing our contributions and our answers to the research questions, pointing out some limitations of our research approach as well as opportunities for further research.

Theoretical Background: CAS Theory and a Framework for Emerging ISD Practices

There is no single and fully shared definition of CAS and unifying CAS theory. The following review of the literature on CAS theory - as our theoretical background - is based on Vidgen and Wang (2009), who have provided a valuable summary on the subject matter, taking into account both the original CAS literature and its applications in managerial, organizational and IS studies. A complex adaptive system consists of a large number of loosely interconnected autonomous parts or agents, each of which behaves according to some set of, sometimes, rather simple rules; these rules require agents to adjust their behavior to that of other agents with whom they interact and adapt to each other. The resulting system behavior can be very complex (Vidgen & Wang 2009, Wang & Conboy 2009). Interaction is a significant concept in this context, as 'the behavior of the system is determined by the nature of these interactions, not by what is contained within the components. Since the interactions are rich, dynamic, nonlinear, and are fed back, the behavior of the system as a whole cannot be predicted from the inspection of its components. The notion of "emergence" is used to delineate this aspect' (Jain & Meso 2004 citing Cilliers 2000). CAS theory rests on the idea that order emerges through the interaction of the agents (Benbya & McKelvey 2006). Emergence is the property of CAS, which creates some greater property of the whole, the system behavior from the interaction of the parts, the agent behaviors; the emergent system behavior cannot be predicted or fully explained from the measured behaviors of the agents (Highsmith 2000). Interaction and emergence are closely related, and link together other generally acknowledged properties of CAS. Beyond the above mentioned interconnected autonomous agents, a number of concepts are frequently used when discussing CAS; these concepts, which we briefly introduce in the following, are: *self-organization, co-evolution, the poise at the edge of chaos, time pacing and the poise at the edge of time*. Interconnected autonomous agents, human or non human, have the ability to independently intervene and determine what action to take, given their perception of their environment; yet, they are interconnected and interact in such a way that they collectively or individually are responsive to change around them, but not overwhelmed by the information flowing to them by this connectivity (Mitleton-Kelly 2003, Vidgen & Wang 2009). *Self-organization* is the capacity of interconnected autonomous agents to evolve into an optimal organized form without external force; it results from the agents' interaction in a disciplined manner within locally defined and followed rules and, as such, requires a departure from command and control management (Volberda & Levin 2003, Vidgen & Wang 2009). *Co-evolution* relates to the fact that a complex adaptive system and/or its parts at a sustainable change rate alter their structures and behaviors in response to the interactions of its parts and to the interaction with other CAS that co-exist in an ecosystem where adaptation by one system affects the other systems; this leads to further adaptations and reciprocal change where the systems do not evolve individually, but concertedly (Kauffman 1993, Vidgen & Wang 2009). *Poise at the edge of chaos* describes the ability of a complex adaptive system to be at the same time stable and unstable, to never quite lock into place, yet never quite fall apart. The edge is the place that provides not only the stimulation and freedom to experiment and adapt as well as for novelty to appear, but it also allows for the sufficient frameworks and structures to avoid disorderly disintegration; this gives a competitive advantage: CAS that are driven to the edge of chaos out-compete those that are not (Brown & Eisenhardt 1998, Anderson 1999, Stacey 2003, McMillian 2004, Vidgen & Wang 2009*). Time pacing* in this context indicates that a complex adaptive system creates an internal rhythm that drives the momentum of change, and that change in CAS is triggered by the passage of time rather than the occurrence of events; this stops them from changing too often or too quickly (Brown & Eisenhardt 1998, Vidgen & Wang 2009). *Poise at the edge of time* conceptualizes a complex adaptive system's attribute of simultaneously being rooted in the present, yet being aware of the future and its balance as well as synchronization between engaging enough exploitation of existing resources and capabilities to ensure current viability with engagement of enough exploration of new opportunities to ensure future viability (Brown & Eisenhardt 1998, Volberda & Levin 2003, Vidgen & Wang 2009). All these core concepts are heavily intertwined and mutually reinforcing (Vidgen & Wang 2006). Thus, CAS can be characterized through the emergence of co-evolutionary, self-organized behavior, structure and order through the interaction of interconnected autonomous agents in a time-paced rhythm balanced at the edge of time. At the heart of CAS theory's relevance for ISD is the concept of emergence (Highsmith 2000), which appears in relation to all key concepts:

(1) as emergent order resulting from self-organizing agent interactions (Kauffman 1993, Holland 1995),
(2) as the emergence of team learning as a result of the interaction and capabilities of interconnected autonomous agents (Mitleton-Kelly 2003) ,
(3) as truly emergent self-organization in contrast to a premeditated or deliberately implemented one by management (Stacey 2003),

(4) as co-evolutionary based emergent behavior and structure (Benbya & McKelvey 2006) or even emergent co-evolution (Kelly 1994),
(5) as the region of emergent complexity, the edge of chaos (Benbya & McKelvey 2006),
(6) as rhythm considered an emergent state of working in the context of time pacing (Vidgen & Wang 2009), and
(7) as emergent balance between exploitation and exploration at the edge of time (Bocanet & Ponsiglione 2012).

In the context of ISD, the emergence of local ISD practices has earlier been researched by Kautz (2004), Madsen et al. (2006), Madsen and Kautz (2009), and Kautz and Madsen (2010), who, for this purpose, developed the emergent method framework, which understands the local practices as organizational innovations (Slappendel 1996), allowing for meticulous mapping and a fine-grained analysis of specific ISD practice. In this context, the emergent method has originally been defined as the unfolding development process and activities, as well as the applied method elements that constitute this process (Madsen & Kautz 2009). This definition addresses the development process as a sequence of activities (Sambamurthy & Kirsch 2000). The choice of the concept of the emergent method was inspired by Pettigrew (1987), who argues that from a systemic perspective, the language of process is characterized by verb forms such as emerging, changing, dissolving and transforming, whereas at the level of the individual actor, the emphasis is on enacting, acting, reacting, interacting, and adapting.

The emergent method framework allows for the identification, presentation and analysis of the applied practices as well as investigation of the unfolding of the actual development process as an outcome of a complex interplay of enacting and interacting actors. This framework emphasizes the significance of organizational structure, the individual project participants' characteristics as well as the relations between the social context and social process, which are tied to the development practices for the emergence of these practices. In order to explain what influences and shapes an emergent method, and how and why it emerges and unfolds in projects the way it does, the framework integrates three perspectives, namely: (1) *the structuralist*, (2) *the individualist* and (3) *the interactive process perspective*, and draws on theoretical ideas as put forward in: Fitzgerald et al.'s (2002) method-in-action framework, Schön's (1983) notion of the reflective practitioner as well as Pettigrew's contextualism (1985, 1987), and Giddens' structuration theory (1984), subsequently synthesized by Walsham (1993). Each perspective supplies a set of key concepts for conceptual understanding and empirical exploration of ISD and ISD method emergence in practice on a detailed level. To set the scene, the structuralist perspective addresses descriptive characteristics of the information system under development, the formalized method, the involved development team and its members, as well as the project's structural context. When studying a process, it is however also necessary to focus on individuals, that is, their demographics, role and perceptions as influential actors, and their social relations and broader social context, as well as the issues of power and culture to generate meaningful explanations (Pentland 1999). Thus, the individualist perspective focuses on the role of the individual project participants' behavior during the formation of the method, their prior methodical and practical knowledge repertoire, their language skills and their preferences of media when communicating project relevant information. In order to explain why the processes unfold as they do, the framework draws on the key concepts of: content of change, social process and social context (Walsham 1993), and focuses on the dynamic aspects of ISD in the interactive process perspective. *Content of change* refers to the planned and actual ISD process and the product of the development endeavor (Kautz 2004), the information system under development. The possible gap between the expected and the actual, if it exists, is important for an initial understanding of what characterizes the content and drives the process of change. The concept of content is therefore applied in order to understand what characterizes the change (Pettigrew 1987). *Social process* focuses on the political, i.e., the distribution of power and balance between autonomy and control, and the cultural - i.e., sub-cultures and the interaction between sub-cultures - aspects of ISD and helps explain how, that is, through which mechanisms, changes to the content take place (Pettigrew 1987, Walsham 1993). *Social context* addresses social relations, social infrastructure and the history of previous procedures, structures and commitments, and helps explain why the social process emerges as it does (Walsham 1993). The social context creates the social and structural landscape within which the social process can emerge, and the social process, in turn, both enables and constrains the content of change (Madsen et al. 2006).

Combining the emergent method framework with its focus on ISD practice with CAS seems reasonable and advantageous when the objective of the research is to investigate the unfolding of the ISD process as an outcome of a complex interplay of enacting and interacting actors. This is because the different interlinked units of analysis facilitate progression from surface description to in-depth interpretation and explanation (Pettigrew 1987, Pentland 1999) of ISD projects as CAS.

Research Approach

Given the limited literature concerning our research topic, and understanding ISD from a CAS perspective, our investigation is based on an exploratory, qualitative, single case study (Creswell 2003) of a contemporary ISD project. While it is often stated that it is not possible to generalize and certainly not to theorize from a single case study, Walsham (1995) suggests that it is possible to generalize case study findings, among others, in the form of a contribution of rich insights. Thus inspired, we have used our theoretical framework as background for our data collection, the coding of the data and the data analysis. Our research is based on an empirical case study of a commercial ISD project in a large German public sector organization, called WaterWorks, performed by a German software company, called IsDev. Upon their request, the names of the organizations have been made anonymous. The research presented is part of a larger project which aims at understanding ISD in practice and at contributing to sustainable theories of ISD. In the context of this paper, we focus on CAS theory as a viable foundation for such theories. For this purpose, the collected empirical data were revisited and in particular analyzed with regard to the structuralist, individualist and interactive process elements that make up the emergent method framework and which shaped the different development activities and practices on the background of the development method, which was applied. The data was then re-interpreted with regard to the key concepts of CAS theory - as identified and discussed in the previous section. The original empirical data for the case study were collected in semi-structured, open-ended interviews conducted by a team of two researchers in a three day period on the development site. The research team performed 12 interviews with 11 individuals (the IsDev project manager was interviewed twice). This included nearly a third of the development team covering project managers and developers with a wide range of experience and skills (for further information on the development team see the section on the case setting), as well as a representative sample of key players and future users in the customer organization. The interviews were tape-recorded and subsequently transcribed. For the coding and qualitative data examination, the text analysis software tool NVIVO7 was used. The interview data was supplemented with company and project documents such as method, requirements and release descriptions, as well as project plans. The ISD project was organized in two phases. At the time that this study was performed, phase one (which had lasted 12 months) had been successfully closed, and phase two had been going on for four months. Responding to an inquiry call during our first data analysis, the responsible project manager stated that phase two ended ten months after our study, on time and budget, and with all parts of the IS being operational. As part of the analysis, we produced an account of our case of ISD in practice, which is structured according to the emergent method framework. This is presented in the next section.

The Case Setting: An Account of ISD in Practice – The OMS Project

The project under investigation was concerned with the development of an operations management system (OMS) for the WaterWorks of a large German city by the software company IsDev. The system was developed with a web-based graphical user interface and a backend to interface the technical infrastructure, as defined by an underlying ERP system. The OMS project was described as a success by both the customer and the development company. We now present the project in greater detail in light of the structuralist, individualist, and interactive process perspectives, and summarize its most prominent characteristics in Tables 1, 2, and 3, respectively.

The Structuralist Perspective

At the time of the project IsDev consisted of about 25 employees, 20 of whom were developers. IsDev based its approach on the development method extreme programming (Beck & Fowler 2001, Beck & Andreas 2004), which provides a number of practices and tools to structure the ISD process. The formalized method includes planning techniques for releases and short iterations called planning games, A5 sized user stories and story cards to specify user requirements, onsite customers to support customer-developer communication, daily stand-up meetings for all project members, pair programming, collective ownership, re-factoring of the developed software, as well as continuous integration and testing to develop the software proper. IsDev extended the method with some project management processes to cater for their projects such as an elaborate overall project plan, formal reporting mechanisms and a formal contract based on an initial requirements specification produced by the customer.

Table 1: Structuralist Perspective

Concepts	Characteristics of and work practices in the OMS Project
Information System	Operations management system (OMS) for the WaterWorks of a city
Formalized Method	XP with short releases/ iterations, planning games , user stories/story cards, onsite users, pair programming, stand-up meetings, working software as measure of progress, short concise documents extended with project management processes/documents, initial project plan, a formal contract, a first requirements specification
Structural Context	Project team onsite in a WaterWorks building, overall project organization with 2-phased development approach, 4 subprojects, sophisticated project organization, limited experience with ISD projects using chosen approach Planning games/ stand-up meetings structured task distribution Pair programming organized individuals to work in shifting developer pairs Stand-up meetings/pair programming processes refined during the project Sophisticated collaboration structure with regular meetings to jointly collect/handle feedback and change requests
Involved Development Team and Developers	Overall 2 overall project managers (1 IsDev, 1 WaterWorks), 12 IsDev staff with multiple roles, project manager, analyst, customer contact, developer, 4 subproject development leaders also as customer contacts, analysts, developers 4 WaterWorks subproject leaders with decision mandate also as user representatives, at least 1 additional user representative for each subproject; not onsite the whole time

The overall development process was a two-phased development approach structured around an initial requirements specification phase and a development phase. Between these two phases, a new formal contract was negotiated based on the original requirements specification. In the 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 customers' organization and their decision to contract IsDev also for the actual development of the OMS. In this main development phase the project was organized into four subprojects to provide IT support ranging from customer management to the maintenance of the sewer system, as well as about 12 IsDev development staff with multiple roles such as project manager, analyst, customer contact, and developer working onsite in a building owned by WaterWorks. The overall project team also consisted of a varying number of users with at least one user representing each of the subprojects. A sophisticated management structure and project organization was established and implemented. In addition to two project managers, each representing one of the companies, it consisted of one subproject manager, also acting as contact person from IsDev and one subproject manager, also acting as onsite-customer from WaterWorks for each individual subproject. According to the WaterWorks project manager, "*To have this counterpart was important to provide a first contact person.*" Each subproject thus had about four developers, including the IsDev subproject manager and at least one WaterWorks representative. The development team was composed of highly educated and motivated young staff, but only the IsDev subproject managers had extensive experience with the applied method and techniques. Their project manager with five years of practice with the method was the most experienced team member, but none of them had ever participated in a project with such a makeup and size.

A first software release was provided after three months with the others to be delivered every three to six months. Each release was organized in iterations of three to six weeks duration, at the time of our investigation each subproject had gone through at least two iterations. Working software was produced story card by story card, which attracted the WaterWorks subproject managers and structured their participation in the development process. The working software provided the measure of progress and the short releases, iterations, and feedback cycles provided the structure for its development. It was presented to the customer on a weekly basis, as an IsDev subproject manager described: "*We really try to show something every week … we make a presentation and demonstrate the software. It is a great way to get fast feedback.*" Documents were also used. The project was performed based on an elaborate project plan and an overarching requirements specification, and for each iteration a number of different, but short and concise documents, such as requirement lists and story cards, were outlined. 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."* Plans and planning structured the ISD process as well. One of the WaterWorks subproject managers emphasized that due to the project setting and complexity "*planning is essential*"; another WaterWorks subproject manager added, "*You need this kind of documents – just because of the complexity of the project*" when arguing for the importance of project plans and requirement specifications in ISD projects. The project had an overall long term project plan developed by the IsDev project manager together with the WaterWorks project manager, and covered a 14 month period that anticipated three to six releases, depending on the subprojects. A more fine-grained plan was developed for the individual iterations which made up a release that was detailed to single weeks. At the beginning of each iteration, the development teams produced and estimated all story cards in team sessions in a planning game. The IsDev staff did most of the work in these games, as - based on the customers' explanations and descriptions - the staff developed the requirements lists and the story cards, but the customer representatives were involved with a decision mandate in the prioritization of the story cards. The planning game and the story cards then offered the devices to perform further planning on the most detailed level for very short periods of time when implementing a story card. The planning games and the stand up meetings were also used to structure the distribution of work tasks. The developers either chose tasks themselves or were assigned tasks by common consent – sometimes on the spot – by their subproject manager without any involvement of the project manager. One subproject manager described this as follows: "*In a stand-up meeting a developer had a good idea and I assigned the task to him. The project manager allows such decisions made by ourselves.*" Stand-up meetings, where all teams participated and informed each other about the status of their work as well as about found solutions and unresolved problems, took place every day. These sessions were originally quite detailed and long, but this practice was later refined, and shorter meetings distinguishing between the overall project and the subprojects were introduced and acknowledged as being very helpful.

Pair programming organized the individuals of the development teams to work in shifting pairs of developers in front of a screen while implementing the requirements written down as user stories on story cards as executable code. The developer teams emerged through spontaneous formations, as one developer put it, *"First I look who is free and then I go and work with him."* Two mechanisms here were important: 1) to regularly shift a partner and 2) to regularly shift possession of the keyboard within a team. The developers found it difficult to find the appropriate synchronization points at which to change a partner in the teams of four developers. No common practice existed. However, they did not follow an overly bureaucratic rule, such as shifting partner every morning regardless of the status of a story card. To avoid both too much red tape and too much disorder, some developers preferred to stay with a partner until a card was closed. *"… changing a partner was always a problem, it still is as changing in the middle of a card seems foolish to me and I don't really like doing it …"* said one developer. However, a subproject leader might intervene if a pair had worked together for too long, say three days. The developers started out with a practice for which one developer exclusively held the keyboard and programmed, while the other watched and sometimes commented. To avoid such situations, a process was introduced where after using a stopwatch for 20 minutes, the keyboard had to switch. This was however abandoned as being too bureaucratic and not fruitful in the work environment. The teams eventually found their own pattern. As one developer commented on the emerging practice, "*We don't do that anymore. It didn't function. Well, now it also functions without any explicit rule."*

Dealing with change was an integral part of the daily activities. Beyond the scheduled meetings, some of the WaterWorks subproject managers turned up at the project nearly on a daily basis, while others came regularly and looked over the shoulders of the developers. IsDev's project manager commented on the customer behavior and formulated the project's overall principle on handling change requests: "*Yesterday* *he said something and today he says something else. Requirements have to be clear at the beginning of an iteration and cannot change right in the middle of it. We are agile but not on a daily or hourly change request rate.*" Thus, different feedback mechanisms were introduced and structures provided for collecting ideas and changing requests. Change requests were brought forward by the customers and users on a short time scale via weekly and bi-weekly feedback sessions, and scheduled weekly meetings for all subproject managers were built into an iteration based on presentations and demonstrations of working software. The customer representatives also regularly performed 'road shows' in the user departments to collect feedback as well as ideas and proposals for improvements. Finally, change requests were detected and collected through the scheduled acceptance test sessions for an iteration or a release with customer representatives. Change requests were, as a rule, dealt with and implemented in the next iteration(s) with the exception of minor changes, which after negotiations and mutual agreement, were implemented as part of the current iteration.

Table 2: Individualist Perspective

Concepts	Characteristics of and work practices in the OMS Project
Skills	WaterWorks staff with education as professional engineers, administrators/clerks, and workmen IsDev staff well schooled in ISD, partly limited experience with applied approach/method, no team member had worked in a project of this scale before
Repertoire and language	Experienced developers' repertoire of development methods and documents as media influenced the process Customer staff in direct and indirect participation and with a consultative, informative and participative role with decision mandate communicated domain knowledge
Media	Preference for direct communication supplemented with written material such as requirement lists, story cards and working software

The Individualist Perspective

The project members consisted largely of two groups of individuals, the IsDev and the WaterWorks personnel. The IsDev staff that held the above described roles was well schooled with regard to ISD. Their skill set and repertoire were impacted by their - at least for some parts of the team - limited prior experience with the applied project management approach and the formalized method and techniques, plus the fact that no team member had worked in an ISD project of this scale before. In this situation, the more experienced developers' repertoire of development techniques as provided by extreme programming as part of the formalized method, and their previous work with more conventional development documents as media influenced the development process.

On the WaterWorks side, there were two distinctively identifiable categories of project participants: onsite staff and further operational staff. The onsite staff consisted of: onsite customers who had a direct participative role with a decision mandate executed whenever necessary as well as other user representatives who were occasionally present onsite as direct participants in consultative and informative roles. The further operational staff were located in the WaterWorks divisions and attended with direct, but mostly indirect, participation in consultative and informative roles during the presentation of working software including trials in the divisions and during the road shows. The onsite customers acted as individual leaders, champions and agents with regard to the chosen development approach. They were intermediaries for the operative staff in the divisions, performing their assignment as true representatives of the operational staff, and their understanding as facilitators and communicators strongly supported the operational staff's pronouncement of opinion and gave them an influential and potent voice. The skills of the WaterWorks staff with regard to ISD varied widely; some had an education as professional engineers, others as administrators and clerks and some as workmen. As one IsDev subproject leader expressed it with regard to their language skills: *"Here we have users, who have to take their working gloves off before they go to the keyboard ... in contrast to projects I've been involved in before, where the customers were wearing ties, here the subproject managers are partly folks, who have done something quite different before, they have a different education and that becomes apparent with regard to their abstraction capabilities and their abilities to write down some texts."*

The differing skill sets had consequences for the choice of preferred media. Despite the number of existing documents, less formal, oral, face-to-face and direct communication supplemented with some written material were the predominant and preferred medium in the OMS project that were perceived by the participants as being sufficient. The initial realization document, the overall requirements specification, although important for the contract and as a starting point for the development process, played a subordinate role; other written media in the form of succinct documents, in particular the requirement lists and more so the story cards were more prominent. These documents were produced, updated, and utilized in the above described way. One developer summarized this as, "*Story cards enable you to understand the context and if you need more details you just ask*"; one of the other developers who shared this perception, confirming that the documents, both in length and in number, were adequate, said: *"I flipped through the realization document in the beginning and never touched it afterwards … ."*

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 IsDev counterpart explained in this context that "*certain users insist on being contacted by phone to be reached straight away while others prefer to be contacted per e-mail, but answer timely.*" The identified media preference for the interaction between the IsDev and the WaterWorks staff could be found throughout the whole course of the project, during the preparation of an iteration in the planning games, during development beyond the scheduled feedback sessions, through the almost daily informal visits and while peeking over the developers' shoulders, and during other presentations and evaluation of the working software in the acceptance tests. The working software and its presentation in the customer performed road shows, as well as the feedback sessions, can thus also be considered as media in the communication process. Finally, intensive oral communication between the IsDev members of the development teams took place during the daily stand-up meetings, during the requirements list and story card production, as well as during the pair programming.

The Interactive Process Perspective

The social context of the OMS project was characterized by relations between IsDev and WaterWorks that had evolved over time from being non-existing to a partnership based on trust and mutual respect. This started in the tendering process, which WaterWorks had opened after a history of several attempts of traditional ISD based on a standard ERP system that had not led to the desired results. As part of the social infrastructure, WaterWorks appointed a project champion, who remembered IsDev's tendering presentation in the following way: "*They presented a method, they explained it, and could convince us to get early user feedback and a working solution.*" Although WaterWorks - based on previous experience - decided to separate the project into two phases with the option to leave after phase 1, the project champion stated: *"… we decided not to be tough on change requests and back-up formalities, but rather to work constructively with them to make progress and my good feelings have been confirmed.*" The WaterWorks Chief IT coordinator also confirmed: *"We have a relationship of trust here.*" The IsDev project leader described the context of requirement changes in the following way: "*The customer is quite relaxed. In such situations they look where they can cut expenses planned for other requirements or we discuss if we can make the implementation simpler to meet the planned budget.*"

The social infrastructure can be portrayed as one in which close collaboration between IsDev and WaterWorks could thrive. To achieve a closer relationship between the teams of IsDev and WaterWorks, the pairing of subproject managers was carefully thought through by the two overall project managers. The WaterWorks project manager commented:"*… it was important for me that they matched with each other.*" In addition, there were always WaterWorks representatives as onsite customers or users, as intermediaries and facilitators (see previous subsection) available, and one WaterWorks subproject manager underlined the commitment to collaboration, stating: "*My management put me here 100%.*" Further forms of collaboration could be identified. Beyond the contacts with the onsite customers, the working software was also presented to larger groups of operational staff. The IsDev project manager summarized the situation: *"Well, at latest when an iteration is finished, sometimes already in the middle of it, or whenever, presentations are run for users. Well, not always in front of many users, but the customer subproject manager gets some people together and says: 'Here, look, do we develop in the right direction?'"* Before an acceptance test was performed, less formal preparations took place often triggered by a WaterWorks subproject manager who would try out the new functionalities described on the story cards, as well as by operational staff end users; thus, it was not just WaterWorks onsite customers who were performing the tests. Finally, the IsDev subproject managers sought direct contact with the operational staff. One manager seated himself for two weeks in the duct operation station with the objective to put the software on trial onsite in order to see how well it fitted the operation. Another IsDev subproject manager engaged some of his developers in this process. After installing a release at one division, they went there several times for some days, discussed with the users and re-built the software accordingly.

Table 3: Interactive Process Perspective

Concepts	Characteristics of and work practices in the OMS Project
Social Context	Relationship WaterWorks - IsDev developed to trustful partnership; in development teams trust, close collaboration, mutual learning prevailed; project manager trusted subproject managers' assigning tasks to project members, subproject managers trusted team members' choice of task, design decisions, choice of programming partner; teams maintained relationships, coordinated and learned through stand-up meetings and pair programming
Social Infrastructure	Project organization supports trust/cooperation; working software presented/evaluated to/by larger groups, informal trials before formal tests, direct contact to operational staff in divisions
Social Process	Culture of openness and collegiality, developers receptive to change, focus on current iteration considering existing software and future extensions; flat hierarchy/autonomy in project/development teams, onsite customers had mandate/authority to decide design matters, software developed with staff and staff council participation
Content of Change	Iterative/collaborative development process, planning/adjustments to plans as reaction to change requests/changes integral project parts; no gap between planned and actual development process IS participatively developed and implemented with negotiated features/architecture

The social context within the IsDev development teams was similarly characterized by trust, close collaboration, and mutual learning. The project manager trusted his subproject managers with regard to assigning tasks to project members, and the subproject managers trusted their team members with regard to choice of task, design decisions and choice of programming partner. The project teams maintained their relationships and coordinated their work through daily stand-up meetings where all developers came together to report briefly about their activities. The stand-up meetings and pair programming were also the means whereby developers learned from each other. One subproject manager described how her pair programming partners contributed to her further development:" *... anyway, he is better than me, and I learn something, and well with the other one I again learn something.*"

This social context of good relations and a supporting social infrastructure had an impact on, and led to, an effective social process. The interaction between the different stakeholder groups went well. The onsite customers sought contact with the operational staff users, the development organization's subproject managers and the rest of the development team, who, in turn, sought contact with the onsite customers and the other users on informal occasions, in feedback meetings, in road shows, during test sessions, and during system operation in the divisions. Beyond mutual trust, respect and collaboration, the project culture was characterized by openness and collegiality. The developers, despite their varying experience, made up a quite homogeneous group, which in general was quite sympathetic and receptive to all customer related issues, including their change requests. The frequent feedback loops had the effect that misunderstandings were caught and dealt with as changes early, that is, before they could grow into something bigger. In each cycle the focus was on the current iteration and on the current user stories, while still taking the existing working software and future extensions into account. An IsDev developer explained: *"Until now it has not happened that everything was totally wrong; there are of course some refinements or a bug is found or something similar. There is always something."* The feedback was taken seriously and responded to with some action within the defined structural context. One developer reported, *"Through the feedback we got, we could react directly ... ."* Politics and the distribution of power - the second trait of a social process - were characterized by a relatively flat hierarchy and autonomy in the project. The development teams, where the onsite customers held a clear mandate with authority to decide on all functional design matters, used it to the satisfaction of all corporative actors.

This led to true participation and controlled a possible dominance of the developers. The working software was always designed with close participation of the WaterWorks subproject managers and other users, and under the mandate of the WaterWorks subproject managers. While the IsDev subproject managers had the liberty to make proposals, the onsite customers could at any time say 'no'; with really important issues, IsDev would get back to the onsite customers before they would go forward. This also included direct contact of those developers who did not act as contact persons or subproject managers with the onsite customers. To support the chosen approach, a project champion was appointed, who strongly lobbied the project. At WaterWorks, there were two further power players who influenced the project: the staff council and the internal IT department. The staff council was perceived as strong political players because they "*can stop the entire project*" - as a WaterWorks subproject manager put it. Involving them in the presentations of the working software and in software testing dispelled their concerns and calmed them down. A WaterWorks subproject manager described this as follows: "*We invited them early to our user workshops and they could observe and already see if they find critical issues*." In addition, some employees of WaterWorks' IT department preferred a particular ERP system in which they had been involved, and pressed the case for this system. A WaterWorks subproject manager remembered: "*They came to me and explained how their system could support my business processes,*" and he added with regard to the two players *"... we have lived with this potential conflict situation since the first day*." These issues did not become dominant; they were dealt with by applying the described measures, which showed the benefits of the approach and the system under development. Handling change requests was one such measure.

The OMS project took place as an iterative and collaborative development process. Planning and adjustments to plans as reaction to change requests and changes were integral parts and, as such, characteristics of the development process. In this sense, while the course of the project was unforeseeable with regard to this aspect of the content of change, no gap between the planned and the actual development process that could provide further insight for an understanding of the process could be identified. The product of the process, the OMS under development to support the operations at WaterWorks, was largely undisputed, negotiated and participatively developed and implemented with the features and the architecture as introduced in this section. We now revisit the characteristics of the presented ISD process by considering the OMS project as a complex adaptive system.

The OMS Project as a Complex Adaptive System: Some Findings

Interactions are a significant characteristic of a complex adaptive system. Within the overall project organization, which had been established by the management of the two project partners and which - consisted of a two-phased development approach, an initial project plan, four subprojects, a formal contract, and a first requirements specification, interactions in the OMS project between the project participants and with other WaterWorks staff took place throughout the whole project in multiple forms, formal and informal, and at numerous occasions - organized and spontaneous. They could be found during the preparations of an iteration in the planning games, in the scheduled weekly subproject manager meetings, the bi-weekly and weekly feedback sessions, during development when clarifying urgent issues face-to-face, on the telephone or via email and through the almost daily informal visits and while watching the developers' work, as well as during other presentations and evaluations of the working software in trials and the acceptance tests. Intensive interaction between the IsDev members of the development teams also took place during the development of the requirements list and story cards, the daily stand-up meetings and during the pair programming. This is ample evidence of the rich, intense, dynamic, and non-linear interactions, which were fed back into the project and determined its performance, indicating that its progression could not be predicted. The concept of emergence describes this characteristic of CAS. In the following, we revisit the OMS project and accentuate the different facets of emergence and key notions of CAS theory to provide a better understanding of contemporary ISD and to derive some insights for the organization of promising ISD projects. Table 4 contains, as summary of our findings, the CAS concepts, their characteristics and the emerging effects in the OMS project.

Interconnected, Autonomous Agents and the Emergence of Team Learning

In the OMS project, both the IsDev and the WaterWorks staff acted as autonomous, interconnected agents. Their autonomy was expressed in numerous ways. The autonomy of the WaterWorks' staff in the project showed when they took part in prioritizing, estimating, designing, and especially in approving design decisions where the onsite customers had a mandate to make decisions and exert power. Autonomy also became apparent in the liberty that the developers had when distributing and picking tasks. Further, when implementing the story cards, the developers acted largely as self-governing with regard to the technical design decisions they made. Finally, the developers were sovereign when choosing a partner for the pair programming. Still, the project participants were highly interlinked and maintained their relationships through the above described structures and measures, which supported various forms of interactions to achieve interconnectivity across and within the four development teams. As noted by Vidgen and Wang (2009), encouraged by the physical presence and closeness in the project facilities onsite at the customer premises, the sharing of project-relevant knowledge in the scheduled weekly feedback sessions and daily stand-up meetings in combination with all team members' involvement in project management, in design decisions and in all development activities, as well as their self-assignment of tasks based on competence and interest, and the rotation of direct collaboration partners in the programming pairs led to the emergence of learning of the entire team.

Table 4: CAS Characteristics of the OMS project

CAS Concept	CAS Characteristics in the OMS project	Emerging Effect
Interaction	Multiple forms of interactions: formal/informal, organi-zed/ spontaneous; in planning games, in scheduled weekly project manager meetings, bi-weekly/ weekly feedback sessions, during development face-to-face, on phone, via email, during presentation/ evaluation of working software; during development of requirements list/story cards, daily stand-up meetings and during pair programming	Rich, intense, dynamic, and non-linear interactions fed back into project and determined its performance; shows that its course could not be predicted
Inter-connected, Autonomous Agents	Interconnected through physical presence and closeness in the project, sharing of project knowledge in scheduled feedback sessions and daily meetings Customer staff autonomy during prioritizing, estima-ting, designing, approving design decisions with onsite customer mandate to make decisions and exert power Developer autonomy when distributing, picking tasks, implementing story cards, making design decisions, choosing pair programming partners	Team members' involvement in project management, design decisions and development activities, self-assignment of tasks, rotation of direct collaboration partners led to the emergence of team learning
Emergent Self-Organization	Self-organization through short communication paths, shared responsibility for governance /decision making, involvement in all activities, task self-assignment Customer self-organization further evidenced through their giving comments in passing, by phone, per e-mail Development staff's self-organization further evidenced through visits to some departments Project managers more like peers/part of the team to primarily act as facilitators to create environment that fosters self-organization.	Development of practices for stand-up meetings and for pair programming illustrates emergence of self-organization of autonomous project members and emergence of order; individual and team discipline are not in conflict with, but vital elements of self-organization
Poise at the Edge of Chaos	Stability through two-phased development approach, formal contract, initial project plan, first requirements specification and the overall implemented project organization Stability and flexible handling of new, demands/ideas through short iterations with frozen requirements, frequent planning, task estimation, prioritization,	Direct reaction on clarifications/ refinements of existing requirements, handling of constant requests for changes/ new requirements illustrate project's maneuvering in a region of emergent

	dirigible task and user story size, stand-up meetings and feedback sessions	complexity
Emergent Co-evolution	Joint learning of customer and development staff as well as software evolution through multiple forms of close interactions and knowledge sharing, continuous gathering/refining of requirements, frequent working software presentations/ delivery and feedback	Mutual learning reinforced emerging structures of collaboration and interaction and behavior of the indivi-dual project participants
Time Pacing	Internally set pace through frequent releases, short and of pre-defined length iterations, formal weekly, bi-weekly meetings and feedback sessions, daily meetings	Appropriate intervals to match/handle changes based on regular planning, re-scheduling; manageable size of requirements supported emergence of working rhythm; most obvious in the way pair programming with regard to shifting partners/ keyboard developed
The Edge of Time	Product focus on today at the same time preparing for future through daily stand-up meetings, pair program-ming, frequent presentations of working software Product focus on past and future through overall project plan, and frequent planning sessions Process focus on present and future through frequent manager meetings and feedback sessions, reflected in flexible, for single iterations fixed, time boxes and developed formats for stand-up meetings and pair programming	Iterations of with existing requirement lists, story card production and writing working software whilst investigating prospective options with roots in the presence and awareness of past and future, balanced concurrent exploitation of existing knowledge and ex-ploration of new knowledge

Emergent Self-Organization and the Emergence of Order

The concept of self-organization is closely related to the concept of interconnected, autonomous agents (Volberda & Levin 2003, Vidgen & Wang 2009). It is thus not unexpected that the same structures and mechanisms, such as short communication paths, shared responsibility for governance and decision making, involvement in all activities as well as task self-assignment, were prominent in the support of self-organization. The described development of how the practices of stand-up meetings and pair programming were first implemented and later refined illustrates the emergence of self-organization of autonomous project members and the subsequent emergence of order in the development process of the OMS project. It also shows that individual and team discipline are not in conflict with, but a vital element of, self-organization. The role of the IsDev subproject managers also deserves some attention. With largely the same development tasks and their sharing of responsibility, they appeared more like peers and a part of the team, who primarily, instead of controlling, acted as facilitators who created an environment that fostered self-organization. In this context, the way feedback - beyond the self-organized regular feedback sessions - was provided and gathered from the customer staff and subsequently handled in orderly form as part of the planning activities also reflects the self-organization of the project members where some customers gave their comments in passing, others by phone, and yet others per e-mail. Finally, the IsDev staff members' visits to some departments indicate a further example of self-organization.

Poise at the Edge of Chaos and the Region of Emergent Complexity

The OMS project was continuously in a state of bounded instability, which means that it – paradoxically – was simultaneously stable and unstable (Stacey 2003). The two-phased development approach, the formal contract, the initial project plan and the first requirements specification, as well as the overall implemented project organization acted as super-ordinate structuring mechanisms that created a relative stable space within which the development process and the working software could unfold. Beyond the direct reaction on manageable clarifications and refinements of existing requirements, the handling of the constant requests for changes and new requirements illustrates how the project maneuvered in a region of emergent complexity, balanced at the edge of chaos. The short iterations with frozen requirements, the frequent planning (including the estimation of tasks with regard to the developers' capabilities and the prioritization of tasks with regard to the actual situation), and not least the dirigible size of the individual tasks and user stories, whose descriptions could not exceed an A5 story card, as well as the stand-up meetings and feedback sessions - all allowed for the flexible handling of the large amount of new, incoming demands and ideas. At the same time, they provided a frame for stability, as they structured the project participants' day-to-day activities and helped them to know what to do, when to do it, and what to expect from others.

Emergent Co-evolution and the Emergence of Behavior and Structure

Co-evolution emerged in the OMS project through the above described multiple forms of close interactions in which the IsDev and the WaterWorks staff shared knowledge and learned from each other. The mutual learning had the reciprocal effect of reinforcing the emerging structures of collaboration and interaction, as well as the behavior of the individual project participants. In particular, the continuous gathering and refining of requirements, along with the frequent presentations and delivery of working software fuelled this process and provided customers with opportunities to learn how to use the OMS, as well as to create new ideas that were then fed back to the development team to become part of the working software. Thus, the working software co-evolved with human actors. Together, this demonstrates the co-evolution of people, processes and products, as well as the unpredictable emergent behavior of the different entities of the CAS during ISD (Meso & Jain 2006).

Time Pacing and the Emergence of Rhythm

In the OMS project, the two-phased development approach and the overall project plan set the time frame for the development process. The frequent releases and, in particular, the iterations, which - although of varying duration - were comparably short and of a pre-defined length, the formal weekly and bi-weekly meetings and feedback sessions, as well as the daily meetings were performed continuously in accordance with an internally set pace. Based on regular planning and, if necessary, rescheduling with regard to releases, iterations and daily activities, this provided the appropriate intervals to match and handle the changes, which were brought up continuously during the project. Together with the manageable size of the requirements on the story cards, it supported the emergence of a lasting working rhythm, which became most obvious in the way that the pair programming both with regard to shifting partners and keyboard developed. In this way, time pacing, as also reported by Vidgen and Wang (2006, 2009), as an organizing principle created an internal rhythm, which drove change in the project in accordance with the passage of time and which, at the same time, allowed for stability and flexibility.

The Edge of Time and the Emergent Balance of Exploitation and Exploration

In the OMS project, the center of attention was always the current iteration and the current user stories while also taking into account the existing working software and the design for future extensions. The iterations of using the existing requirement lists to produce story cards and write working software whilst investigating prospective options with information from, and in consultation with customer staff (with its root in the presence and simultaneous awareness of the past and the future) balanced the concurrent exploitation of existing knowledge and the exploration of new knowledge. This has been characterized as the edge of time by Brown and Eisenhardt (1998). The daily stand-up meetings and the pair programming served the purpose of keeping a focus on today, and, at the same time, as preparing for the future. This is also true for the frequent presentations of working software, while the overall project plan, as well as the frequent planning sessions structured around releases, iterations, planning games and the implementation of working software, supported a focus on, and constituted a manifestation of, both the past and the future. Beyond product-oriented exploitation and exploration and providing the opportunity for direct reaction, the frequent manager meetings and feedback sessions were also used by the project participants to think about their own behavior and to review and improve the development process. This is reflected in the flexible (but for an iteration fixed) time boxes and the developed formats for the stand-up meetings and the pair programming.

Discussion

CAS theory has been recognized as a valuable approach to understanding contemporary ISD, and several research teams have applied the theory and provided organizing principles and suggestions for best practice for the ISD process. We discuss our research in relation to this work.

Benbya and McKelvey (2006) present some conceptual work, focusing on the co-evolutionary aspect of CAS, and put forward seven sometimes rather abstract principles of adaptive success, after which the authors discuss their bearings on ISD problems. Accordingly, their advice is quite abstract and, in parts, not easy to implement. Among the principles are: the principle of adaptive tension relating to the dynamic nature of the development process and the tensions caused by contradictory requirements, the principle of requisite complexity relating to the unpredictable nature of future requirements, as well as the principle of causal intricacy relating to the dynamic and continually shifting links between institutional, business and technical requirements. Meso and Jain (2006) identify the seven principles of CAS, which, to some extent, overlap with those described by Benbya and McKelvey (2006), but whose meanings are more intuitive and easier to grasp. They are the principles of: open systems, interactions and relationships, transformative feedback loops, emergent behavior, distributed control, shallow structure, and growth and evolution. The authors provide a largely conceptual argument that they support with various, partly unrelated, examples from the literature. They map the principles to agile development practices, and from this mapping derive more concrete, yet generic recommendations for best practices for ISD. In brief, their recommendations are: (1) Let actual solutions, processes, work patterns, team configurations and interactions emerge naturally rather than from heavy planning and design. (2) Do minimal planning where and when necessary; limit emphasis on documentation. (3) Allow for manageable experimentation in product design and with processes for learning from mistakes. (4) Develop, test and validate the IS iteratively and compare with current and past solutions. (5) Allow solutions to be responsive to emerging changes in requirements by taking into account feedback from frequent releases. (6) Strive for componentization and loosely coupled IS. (7) Allow the development process to emerge or be determined by local needs. (8) Use an iterative, time-boxed, and modular development process with measurable process milestones. (9) Reevaluate the development process and methodology frequently. (10) Delegate responsibility and decision making to local development units. (11) Involve stakeholders and fellow developers by listening to their comments. (12) Reevaluate team configuration frequently and allow for pairing or teaming up of developers. Vidgen and Wang (2009) - again with a focus on co-evolution and agile development - develop a framework grounded in the work of Volberda and Levin (2003) on co-evolving self-organizing organizations. Vidgen and Wang (2009) summarize the key concepts of CAS in three guiding principles for the organization of agile software development as matching co-evolutionary change rate, optimizing self-organization, and synchronizing concurrent exploitation and exploration. Based on an actual and sound empirical case, they identify six capabilities of agile teams, namely, co-evolution of IT team and customer to create business value; sustainable working with rhythm, collective mindfulness, sharing and team learning; process adaptation and improvement; and product innovation. They find practices akin to those posited by Benbya and McKelvey (2006), but consider them as enablers of agility. Correspondingly, they also present inhibitors for agility.

Our work is based on these predecessors, but rather than organizing principles or focusing on particular concepts, we have chosen, as a starting point for our analysis, a set of concepts that characterize CAS: interaction, emergence, self-organization, co-evolution, poise at the edge of chaos, time pacing and poise at the edge of time. Our investigation shows how these concepts are interrelated and gives a broad and detailed examination of the researched case based on the emergent method framework used for the presentation of our case. While other authors either derive abstract principles for the organization of ISD from CAS - which they do not back up with empirical evidence (Benbya & McKelvey 2006) or provide a conceptual argument though less abstract but limited to agile development (Meso & Jain 2006) or focus entirely on agile development with their empirical and conceptual work (Vidgen & Wang 2009) - our main contribution is twofold: We show how CAS and CAS principles are advantageous for comprehending and organizing ISD in general, beyond any particular development approach chosen for the execution of a project such as agile development. Thereby, we contribute to a complexity theory of ISD. Second, we back up our argument with a coherent empirical account of contemporary ISD, and thus our contribution of practical advice for ISD derived from this perspective to successfully cope with complexity in ISD in an adaptive manner. The practices we found are similar to those depicted by Meso and Jain (2006) and Vidgen and Wang (2009). In line with these authors, we consider them to be vital and recommend them. However, beyond confirming many of the earlier presented findings, we extend the line of reasoning and argue that rather than being enablers for agility, these practices are properties of complex, adaptive systems. If ISD is understood as CAS, certain characteristics of the process are recognized to facilitate good performance while others inhibit it. We put forward that they are not limited to, and crucial only for, agile development methods, but that they are important for contemporary ISD in general, which concerns turbulent and changing environments. Our focus is on the ISD process as such - not on a particular approach or method which has to be adapted – in contrast to work which takes as a starting point an agile method and asks how it can be adapted in different contexts (Cao et al. 2009). We have thus so far not particularly emphasized that, as the presentation of our case shows, the development approach of the OMS project was founded on a combination of plan-based and agile practices. Based on our empirical data, CAS theory sheds new light on the discussion concerning the advantages and disadvantages of mixing plan-driven and agile approaches, methods and practices (Abrahamsson et al. 2003, Boehm & Turner 2004, Port & Bui 2009), and goes beyond actual problems and practices currently coined in the literature in order to form a deeper theoretical understanding of ISD (Kautz et al. 2007). While such an integration based on such simple classifications from a pragmatic perspective might be important and is supported by CAS, it seems to be based on a false dichotomic split between agile and other methods (Boehm & Turner 2003), thus appearing to be theoretically secondary and less relevant. It might even render redundant the plan-based versus agile approaches debate as an ‘either-or’ matter (Austin & Devin 2009), as long as the chosen approach allows for organizing for adaptation and adaptive ISD in dynamic settings based on a set of simple foundational rules. In the OMS project, the overall plans and requirements documents, as well as the possibility to change, revise, and refine these plans and requirements, create conditions in which adaptive activities can flourish; as such, the OMS project is not plan-driven, but planning driven. In line with Vidgen and Wang (2009), we find that recurrent planning is a consequence of frequent feedback due to the close relationships and the interactions among all those involved in the project and that both high-level, sketchy, long-term plans as well as detailed, accurate, short-term plans are necessary and beneficial in such a process.

Vidgen and Wang (2009), in the context of self-organization, also identify collective mindfulness as a capability of an agile team; Matook and Kautz (2008) take this a step further, demonstrating the relationship between individual and collective mindfulness, as well as agile development, and showing how mindful behavior may contribute to successful ISD. Butler and Gray (2006) suggest that individual and collective mindfulness increases organizations’ ability to perform in problem solving situations and dynamic, unstable environments such as ISD. According to these authors, “Collective mindfulness requires organizations to couple the ability to quickly detect issues, problems, or opportunities with the power to make organizationally significant decisions. This may be accomplished by moving decision-making authority or creating an organizational environment that enables the smooth interaction of perception and action.” These are characteristics also associated with ISD, that is, when understood as CAS beyond the application of an agile development method, as we demonstrated with our case. In collective mindfulness, existing expectations are continuously scrutinized and refined according to new experiences in order to be able to invent new expectations for dealing with unprecedented situations to improve foresight and current functioning (Weick & Sutcliffe 2001). Five key aspects characterize collective mindfulness: preoccupation with failure, reluctance to simplify, attention to operations, commitment to resilience, and migration of decisions to expertise (Weick & Sutcliffe 2001). At the individual level, Langer (1997) distinguishes five constituent components of mindfulness: openness to novelty, alertness to distinction, sensitivity to different contexts, awareness of multiple perspectives and orientation in the present. In the OMS project, the behavior that led to the emergence of rhythm in the stand-up meetings and the pair programming - in accordance with self-organization and autonomy while balancing at the edge of chaos and time - exhibits both characteristics of collective and individual behavior in the form of alertness to distinction, sensitivity to different contexts, awareness of multiple perspectives, orientation to the present as well a willingness to learn from errors, an aspiration to perceive problems from different perspectives, an ability to cope with problems and the migration of decision to expertise. Thus, we provide further empirical evidence on the role of mindfulness in coping with ISD as

CAS, which can be extended beyond its relationship to self-organization to other concepts of CAS theory. We contend that mindfulness, further than agile development, is a valuable concept to understand and organize contemporary ISD in general.

Conclusion

This paper starts out with the premise that contemporary ISD is a complex activity. On this premise, we investigate the question of whether CAS theory is relevant as a theoretical foundation for understanding ISD, and if so, what kind of understanding can be achieved by utilizing the theory. By doing so, we follow Baskerville et al. (2011), who argue that in a post-agility era of ISD, further theories are needed to understand contemporary ISD. We demonstrate the strength of a CAS approach, showing that CAS is useful, as it directs the focus on characteristics and practices of the development process, which by organizing and facilitating for adaptation, might lead to thriving ISD projects. Based on the application of the emergent method framework (Kautz 2004, Madsen et al. 2006, Madsen & Kautz 2009, Kautz & Madsen 2010) and grounding explanations in CAS theory, our presentation and examination of an empirical case study provide an argument for why and how - despite some identified challenges - project staff successfully carried out an ISD project. Our investigation illustrates that CAS theory is valuable not only for understanding the so-called agile development methodologies and agile development, but independent of a particular methodological approach chosen for the execution of a project, more general for comprehending contemporary ISD. We identify a number of organizing principles, practices and capabilities that support the satisfactory execution of an ISD project. We contend that these traits are not limited to agile development methods, but from a CAS perspective, they are valid in general for contemporary ISD, which deals with turbulent and changing environments. Agile development is only one way of dealing with present day ISD. Further, we argue that when studying and discussing ISD to gain in-depth insight, it is not that relevant to consider whether an applied practice belongs to a specific, agile or traditional, 'school' or a combination of these two; rather, what is more important is how it is actually used in the particular case. Our analysis reveals how a successful ISD project with particular structural characteristics in interplay with the involved individuals unfolds over time. It shows how individuals - as interconnected, but autonomous agents - organize themselves and co-evolve together with the information system under development in the course of the process. It also shows how in this complex web of mutual influences the social context and process impact on how the information system in question is developed in a sustainable pace of time by poising at the edge of time and chaos. Thus, while our work contributes to a complexity theory of ISD, the case examination also provides practical advice to successfully deal with complexity in ISD in an adaptive manner.

In conclusion, we recognize the limitation of our research being based on a single case study. While it is possible to generalize case study findings in the form of a contribution to rich insight (Walsham 1995), further claims concerning generalizability of the findings cannot be made; as an exploratory study, our research provides a foundation for establishing a complexity theory as a viable theory of ISD, and calls for further research efforts concerning such a theory. In this context, the above briefly discussed relationship between complexity and mindfulness in ISD deserves attention in future research. Another open question is the issue of funding processes and the impact of pricing structures on ISD practice in relation to control and flexibility in environments that acknowledge unpredictable change and complexity. While they have been discussed and studied in the context of agile development methods (Highsmith 2002, Cao et al. 2012), they have not yet been investigated from a general complexity theory perspective on ISD, thus providing another worthwhile avenue for further research.

Acknowledgements

I would like to thank Ralf Klischewski as a member of the original interview team as well as Sabine Matook (née Zumpe) and Sabine Madsen, who collaborated with me on parts of the data analysis and on reporting other results from the project. The reviewers of the original ICIS submission provided valuable feedback, which helped clarifying the relation between the empirical findings and their theoretical grounding. I am also grateful to the members of the OMS project for their time and insights.

References

Anderson, P. 1999. "Complexity Theory and Organization Science," *Organization Science (10:3), pp*. 216–232.

Abrahamsson, P., Warsta, J., Siponen, M. T. and Ronkainen, J. 2003. "New Directions on Agile Methods: A Comparative Analysis", in *Proceedings of the 25th IEEE International Conference on Software Engineering*, Portland, Oregon, pp. 244-254.

Augustine, S., Payne, B., Sencindiver, F., and Woodcock, S. 2005. "Agile project management: steering from the edges," *Communications of the ACM (48:12)*, pp. 85-89.

Austin, R. D., and Devin, L. 2009. "Research Commentary—Weighing the Benefits and Costs of Flexibility in Making Software: Toward a Contingency Theory of the Determinants of Development Process Design*," Information Systems Research (20:2)*, pp. 462-477.

Baskerville, R., Pries-Heje, J., Madsen, S. 2011. "Post-agility: What follows a decade of agility?," *Information and Software Technology (53)*, pp. 543-555.

Beck, K., and Fowler, M. 2001. *Planning Extreme Programming* (2nd ed.), Boston, MA: Addison-Wesley.

Beck, K., and Andreas, C. 2004. *Extreme Programming Explained: Embrace Change* (2nd ed.), Boston, MA: Addison-Wesley.

Benbya, H., and McKelvey, B. 2006. "Toward a complexity theory of information systems development," *Information Technology & People (19:1)*, pp. 12-34.

Bocanet, A., and Ponsiglione, C. 2012. "Balancing exploration and exploitation in complex environments," *VINE - The Journal of Information and Knowledge Management Systems* (42:1), pp. 15-35.

Boehm, B., and Turner, R. 2003. "Using Risk to Balance Agile and Plan Driven Methods," *Computer (36:6)*, pp. 57-66.

Boehm, B., and Turner, R. 2004. "Balancing agility and discipline: evaluating and integrating agile and plan-driven methods," in *Proceedings of the 26th IEEE International Conference on Software Engineering*, Washington DC, pp. 718–719.

Brown, S., and Eisenhardt, K. 1998. *Competing on the Edge: Strategy as Structured Chaos*. Boston, MA: Harvard Business School Press.

Butler, B. S., and Gray, P. H. 2006. "Reliability, Mindfulness and Information Systems", *MIS Quarterly (30:2)*, pp 211-224.

Cao, L., Mohan, K., Xu, P., and Ramesh, B. 2009. "A framework for adapting agile development methodologies," *European Journal of Information Systems (18:4)*, pp. 332-343.

Cao, L., Mohan, K., Ramesh, B., and Sarkar, S. 2012. "Adapting funding processes for agile IT projects: an empirical investigation," *European Journal of Information Systems (21:1)*, pp. 1-15.

Cilliers, P. 1998. *Complexity and Postmodernism: Understanding Complex Systems*, London, UK: Routledge.

Cilliers, P. 2000. "What Can We Learn From a Theory of Complexity?," *Emergence (2:1)*, pp. 23-33.

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

Desai, A. 2005. "Introduction into special section on Adaptive Complex Enterprises," *Communications of the ACM* (48:5), pp. 32-35.

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

Giddens, A. 1984. *The Constitution of Society*, Cambridge, UK: Polity Press.

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

Highsmith, J. 2002. *Agile Software Development Ecosystems*, Boston, MA, USA: Addison-Wesley, Pearson Education.

Holland, J.H. 1995. *Hidden Order: How Adaptation Builds Complexity*, Reading, MA: Addison-Wesley.

Holland, J.H. 1998. *Emergence: From Chaos to Order*, Cambridge, MA: Perseus Publishing.

Jacucci, E., Hanseth, O., Lyytinen, K. 2006. "Taking complexity seriously in is research. Introduction to the Special Issue," *Information Technology & People (19:1)*, pp. 5-11.

Jain, R., and Meso, P. 2004. "Theory of Complex Adaptive Systems and Agile Software Development," in *Proceedings of the 10th Americas Conference on Information Systems*, New York, NY, pp. 1661-1668.

Kauffman, S. 1993. *The Origins of Order: Self-Organization and Selection in Evolution*, New York, NY: Oxford University Press.

Kauffman, S. 1995. *At Home in the Universe*, New York, NY: Oxford University Press.

Kautz, K. 2004. "The Enactment of Methodology: The Case of Developing a Multimedia Information System," in *Proceedings of the International Conference on Information Systems*, Washington, DC, USA, pp. 671-684.

Kautz, K., Madsen, S., Nørbjerg, J. 2007. "Persistent Problems and Practices in Information System Development," *Information Systems Journal (17:3)*, pp. 217-239.

Kautz, K., and Zumpe, S. 2008. "Just Enough Structure at the Edge of Chaos: Agile Information Systems Development in Practice," in *Proceedings of the International Conference XP 2008*, Limerick, Ireland. Abrahamsson, P. et al. (eds.), Berlin, Germany: Springer Notes in Business Information Processing, pp. 137-146.

Kautz, K., and Madsen, S. 2010. "Understanding Agile Software Development in Practice," in *Proceedings of the 2010 International Conference on Information Resources Management*, Rose Bay, Jamaica.

Kelly, K. 1994. *Out of Control - The New Biology of Machines, Social Systems, and the Economic World*, New York, NY: Addison-Wesley.

Langer, E.J. 1997. *The Power of Mindful Learning,* Cambridge MA: Perseus Publishing.

Madsen, S., and Kautz, K. 2009. "A Framework for Identifying the Drivers of ISD Method Emergence," in *Systems Analysis and Design: Techniques, Methodologies, Approaches, and Architecture*. Roger Chiang, R., Siau, S., Hardgrave, B. C. (eds.), Armonk, NY: M.E. Sharpe, pp. 58-71.

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

Matook, S., and Kautz, K. 2008. "Mindfulness and Agile Software Development," in *Proceedings of the 19th Australasian Conference on Information*, Christchurch, NZ, pp. 638-647.

McMillan, E. 2004. *Complexity, Organizations, and Change*, London, UK: Routledge.

Merali, Y., and McKelvey, B. 2006. "Using complexity science to effect a paradigm shift in information systems for the 21st century," *Journal of Information Technology (*21:4), pp. 211-215.

Meso, P. and Jain, R. 2006. "Agile Software Development: Adaptive Systems Principles and Best Practices," *Information Systems Management (23:3)*, pp.19-30.

Mitleton-Kelly, E. 2003. "Ten principles of complexity and enabling infrastructures," In *Complex systems and evolutionary perspectives on organisations: the application of complexity theory to organisations*. Mitleton-Kelly, E. (ed.) Oxford, UK: Elsevier Science Ltd, pp. 3-20.

Pettigrew A.M. 1987. "Context and action in the transformation of the firm," *Journal of Management Studies (24:6)*, pp. 649-670.

Pettigrew A.M. 1985. "Contextualist research and the study of organisational change processes," in *Research Methods in Information Systems*, Mumford E. et al. (Eds.), Amsterdam, the Netherlands: North-Holland Elsevier Science Publishers, pp. 53-78.

Pentland B.T. 1999. "Building process theory with narrative: from description to explanation," *Academy of Management Review (24:4),* pp. 711-724.

Port, D., and Bui, T. 2009. "Simulating mixed agile and plan-based requirements prioritization strategies: proof-of-concept and practical implications," *European Journal of Information Systems* (18:4), pp. 317-331.

Sambamurthy V., and Kirsch L.J. 2000. "An integrative framework of the information systems development process," *Decision Sciences (31:2) , pp. 391-411.*

Schön, D. A. 1983. *The Reflective Practitioner - How Professionals Think in Action*, New York, NY: Basic Books.

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

Stacey, R. D. 2003. *Strategic Management and Organisational Dynamics: The Challenge of Complexity* (4th ed.), Harlow, UK : Financial Times, Prentice Hall.

Truex, D.P., Baskerville, R., Klein, H. 1999. "Growing Systems in Emergent Organizations," *Communications of the ACM (42:8)*, pp. 117-123.

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

Walsham, G. 1995. "Interpretive Case Studies in IS Research: Nature and Method", *European Journal of Information Systems (4:2)*, pp. 74-81.

Wang, X., and Vidgen, R. 2007. "Order and Chaos in Software Development: A comparison of two software development teams in a major IT company," in *Proceedings of the 16th European Conference* *on Information Systems*, Winter, R., et al. (eds.), St. Gallen, Switzerland , pp. 807-818.

Wang, X., and Conboy, K. 2009. "Understanding Agility in Software Development through a Complex Adaptive Systems Perspective," in *Proceedings of the 17th European Conference on Information Systems*, Verona, Italy.

Weick, K., and Sutcliffe, K. H. 2001. *Managing the Unexpected: Assuring High Performance in an Age of Complexity*, San Francisco, CA: Jossey-Bass.

Vidgen, R., and Wang, X. 2006. "Organizing for Agility: A Complex Adaptive Systems Perspective on Agile Software Development Process," in *Proceedings of the 14th European Conference on Information Systems*, Goeteborg, Sweden. Ljunberg J., and Andersson, M. (eds.), pp. 1316-1327.

Vidgen, R., and Wang, X. 2009. "Coevolving Systems and the Organization of Agile Software Development," *Information Systems Research (20:3)*, pp. 355-376.

Volberda, H. W., and Levin. A. Y. 2003. "Guest editors' introduction: coevolutionary dynamics within and between firms: From evolution to co-evolution," *Journal of Management Studies (*40:8), pp. 2111-2136.