Master Thesis in
Software Engineering
Thesis no: MSE-2004:01
January 2004

Thriving at the Edge of Chaos

— An Investigation of Applying Complexity Theory on Software Organizations

Jonas Bengtsson

Department of
Software Engineering and Computer Science
Blekinge Institute of Technology
Box 520
SE - 372 25 Ronneby
Sweden

Contact information:
Author:
Jonas Bengtsson
Address:
Knut Hahnsgatan 7B
SE – 372 35 Ronneby
E-mail:
jonas.b@home.se

University advisors:
Conny Johansson
Department of Software Engineering and Computer Science

Anders Nilsson
Department of Business Administration and Social Science

Department of
Software Engineering and Computer Science
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby

Internet	: www.bth.se/ipd
Phone	: +46 457 38 50 00
Fax	: + 46 457 271 25

ABSTRACT

In this master thesis two different worldviews are compared: a mechanistic, and an organic worldview. The way we think the world and the nature work reflects on how we think organizations work, or how they ought to work. The mechanistic worldview has dominated our way of thinking since the seventeenth century, and it compares the world with a machine. The organic worldview could use a number of different metaphors, but the one addressed in this thesis is complexity theory.

Complexity theory is related to chaos theory and is concerned with complex adaptive systems (cas). Complex adaptive systems exist everywhere and are systems such as the human immune system, economies, and ecosystems. What complexity theory tries to do is to understand these systems—how they arise, how they function and how order emerge in them. When looking at complex adaptive systems you can't just look at the different parts. You must take a more holistic view and look at the whole and the interaction of the parts. If you just look at the parts you will miss the emergent properties that have emerged as the system has self-organized. One prominent aspect of these systems is that they don't have any central authority, but somehow order do arise. In relation to organizations, complexity theory has something to say about almost all aspects of organizations: from what kind of leadership is needed, and how teams should be organized to the physical structure of the organization.

To understand what complexity theory is and how to relate that to (software developing) organizations is the main focus of this thesis.

Scrum is an agile and lightweight process which can be applied on development projects in general, but have been used in such diverse examples as software development projects, marketing programs, and business process reengineering (BPR) initiatives. In this thesis Scrum is used as an example of how to apply complexity theory to organizations.

The result of the thesis showed that Scrum is highly influenced and compatible with complexity theory, which implies that complexity theory is of some use in software development. However, there are more work to be done to determine how effective it is, how to introduce it into organizations, and to explore more specific implementations.

This master thesis should give the reader a good understanding of what complexity theory is, some specific issues to consider when applying complexity theory on organizations, and some specific examples of how to apply complexity theory on organizations.

Keywords: complexity theory, Scrum, agile software development, organization theory

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

This chapter is an introduction to the thesis. It will start by providing some background information to show the motivation behind this thesis and also a short description of what complexity theory is. After the background the hypothesis will be presented. The succeeding section presents the main purpose of this thesis. At the end of this chapter some reading instructions and a road map for the rest of the thesis will be provided.

Background

For a long time I have been fascinated of how structure arise in a group of people. Especially when it comes to the difference between the formal and the informal structure. It seems to me that there is usually a tendency to try to impose a structure at an early stage of the group's forming. One example is the project in which I participated during my education. We were only sixteen persons and were given a rough outline of what the customer needed. But before we had got to know each other as a group and met the customer, we started creating a structure—assigning six different roles and dividing us into three teams with one team leader per group. We imposed a quite impressive structure on our relatively small group before we knew the specifics of the problems we were going to solve (besides the rough outline) and how we functioned as a group. After a while this structure became outdated—it did not fit the way of working that we thought was the most appropriate, and the actual way of working did not match the official structure—thus we had to impose another structure. However, I have increasingly come to realize that our approach to start imposing a structure on a new group is not optimal. Instead we might had been better off by letting a structure emerge as the work progressed and as we got to know each other as a group. If the first thing you do is to create a structure, you are probably creating it for its own sake and not as a means to an end. And since your understanding of the problem at hand is usually quite limited in the beginning, I suspect that there is quite a large risk that the structure is not that good at facilitating the work and might not suit the way the current group functions.

Another aspect of the difference between formal and informal structures is that there, in my experience, often exists conflicts between them, especially when it comes to roles. For instance, it is not always the appointed leader who leads.

This fascination of the relation and conflict between formality and informality led me to look at more informal approaches of organizing. One of these approaches I came in contact with was Scrum. Scrum is an agile and lightweight process which can be applied on development projects in general, but have been used in such diverse examples as software development projects, marketing programs, and business process reengineering (BPR) initiatives. Through Scrum, and Ken Schwaber (one of the minds behind Scrum), I started looking into the subject of complexity theory.

To give a short introduction to complexity theory (the subject will be discussed more in-depth throughout this thesis), it is related to chaos theory and is concerned with complex adaptive systems (cas). Complex adaptive systems exist everywhere and are systems such as the human immune system, economies, and ecosystems. What complexity theory tries to do is to understand these systems—how they arise, how they function and how order emerge in them. When looking at complex adaptive systems you can't just look at the different parts. You must take a more holistic view and look at the whole and the interaction of the parts. If you just look at the parts you will miss the emergent properties that have emerged as the system has self-organized. One prominent aspect of these systems is that they don't have any central authority, but somehow order do arise.

Metaphors

Metaphors help us to see upon certain things in different, and possibly new, ways. By using metaphors we can compare things, which are not normally considered equal, and transfer properties from one to another. This gives us a tool to get a better understanding of the element we are studying, by comparing it to an element, which we have a better understanding of. For instance, "the man is a lion" makes you apply your perception of a lion (perhaps its strength and courage) to the man in question.

Metaphors have, however, limitations since they only focus on identifying similarities (Hatch, 1997). By only looking for similarities you are disregarding, or at least downplaying, the differences between the things. If you don't take the differences into account, you can start "overextending the metaphor by taking it to ridiculous extremes." (p. 55) Because of these limitations, metaphors can never give a full understanding of what you are studying.

Metaphors are not limited to just work as a tool we choose to use when we compare different things. They are more influential than that. "For the use of metaphor implies *a way of thinking* and *a way of seeing* that pervade how we understand our world generally. . . . [M]etaphor exerts a formative influence on science, on our language and how we think, as well as on how we express ourselves on a day-to-day basis." (Morgan, 1986, pp. 12-13)

A good way to understand organizations is to use metaphors since organizations are "complex and paradoxical phenomena that can be understood in many ways." (Morgan, 1986, p. 13) Thus, by using metaphors on organizations we get new perspectives and might discover new ways of organizing. In this thesis the emphasis is on two different metaphors, and their corresponding worldviews, namely the machine metaphor and the organic metaphor (or more specifically the metaphor of complex adaptive systems).

Software Industry

Software developing organizations live in a complex and chaotic environment. This is clearly manifested by the findings of the CHAOS study by The Standish Group (1998), which show that only 26% of the projects were successful (completed on time and budget, with all the functionality originally specified), 46%

were challenged (completed but over time and budget, with less functionality than originally specified) and 28% were cancelled before completion. This low success rate suggests that software organizations have problems managing their complex environment. The study is also suggesting that the success rate for small projects is higher than for large projects, due to the higher complexity, confusion and cost—the larger projects are experiencing more of the complex environment. This state of the software industry is concisely captured in the expression software crisis, which was primarily used in the 1960's and 1970's, and is referring to the delivery and quality problems of the software industry.

But why is the software industry such complex business? First of all, it is not unique for the software organizations—most organizations live in a complex environment. Nowadays organizations are not just competing on a local market, but due to the globalization, where Internet has played a significant role, they are often competing on an international market. The markets have grown, and so has the number of competitors and customers.

Another reason for the complexity is the ever-changing technology, which software organizations are highly dependent upon. There are a lot of development and inventions both in hardware and software, to which the organizations must be up to date.

Changing requirements have a strong influence on the complexity of the projects. As the customers learn more about what need they want the application to fulfil, they get new requirements. If the projects want to satisfy their customers and respond well to their changing needs, they can't expect that what has been decided upon earlier and perhaps already delivered is still valid.

Because of the high level of complexity on organizational level as well as project level, software organizations must be flexible. To survive in a turbulent environment you must adapt to be able to react to the new demands you are exposed to.

Hypothesis

Complexity theory contains elements and aspects that can be applied on software organizations—it is a meaningful metaphor. By studying complex adaptive systems it is possible to derive new ways of organizing software organizations in a way that better adapts to the ever-changing environment than current, mechanistic, organizations. By emphasizing self-organization, autonomy, leadership and collaboration it is possible to create organizations that are well suited to live in today's complex world.

Purpose

The primary purpose of this thesis is to explore what we can learn about organizing software organizations using complexity theory as a metaphor.

This will be realized through a study of what complexity theory implies. The subject is an emerging discipline, which is not nearly as mature as other disciplines, such as mathematics or mechanics. Complexity theory is also quite complex since it tries to take a holistic and interdisciplinary approach. Hence, to aim at getting and providing a comprehensive view of an emerging and complex discipline is not realistic within the bounds of this thesis. Instead I will aim at providing the view that corresponds to the view I have acquired through studying literature of various types and with various perspectives. When choosing the literature I have tried to get a fair representation of the subject in order to be able to provide a not too colored view.

Complexity theory, and the organic worldview in general, will also be compared with the mechanistic worldview to see in what ways they differ. By comparing the mechanistic worldview, which we are more accustomed to, with the organic worldview, the specifics of the organic view will become more apparent.

Method

The hypothesis will be further explored by studying an example of how to apply complexity theory to organizations. The example chosen is the agile methodology Scrum. The exploration of the example will show both to which extent Scrum adheres to complexity theory, and a much more specific application of complexity theory to software development. If there is a good match between Scrum and complexity theory, it will show that complexity theory is of at least some use in software development.

Reading guidelines

Although this thesis investigates how complexity theory can be applied on organizations developing software, very little of the content requires any extensive insight into the software industry. Most of the content is probably applicable on any organization that resembles software organizations—this is however not investigated. Consequently, this thesis should be understandable by most people, in spite of the fact that it is intended for people within the software industry.

Road map

This thesis will start with a map of complexity theory. To provide a foundation for this map two different worldviews will be discussed, the mechanistic and the organic view, where complexity theory is part of the latter. Next, a few aspects of organization will be investigated in the light of complexity theory. Then the example of Scrum will follow, which will give some pragmatic examples of how complexity theory can be used.

Chapter 2—*The Mechanistic View*—presents the view that has dominated our way of thinking since the seventeenth century. This view considers the universe, as well as organizations, to work as machines. The management style that originates from the mechanistic view is command-and-control, and it has a strong central authority. Implications of the mechanistic view can be observed in the software process improvement movement.

Chapter 3—*The Organic View*—presents a view that is more organic and originates from the *new science*—chaos theory and complexity theory. This view has grown strong within certain fields of the science but not in society. The order in this view is not static, but rather dynamic—an orderly disorder. Organizations are not seen as machines but as something organic—as complex adaptive systems. The management style in this view is leadership-and-collaboration, that doesn't have a strong central authority.

Chapter 4—*Organization*—discusses three different aspects of organizations: the social structure, the physical structure, and knowledge management. The social and the physical structure are important to consider in terms of how they effect the self-organization. Knowledge management is also important to consider in an organizational context and it is important that complexity theory enables knowledge creation to take place.

Chapter 5—*Scrum*—explores Scrum and its relation to complexity theory. Scrum is an agile methodology, primarily used for developing software. There is a strong emphasis on commitment on all levels, and the team is autonomous and self-organizes to do the work. During the 30 day iteration the team only has to focus on the work tasks for that iteration. Complexity theory has influenced Scrum, and is a helpful tool to understand how and why it can work.

Chapter 6—*Conclusion*—concludes this thesis, discusses some various issues and areas about the thesis, and looks at what further work is needed.

CHAPTER 2

THE MECHANISTIC VIEW

In this chapter the mechanistic view will be described. The mechanistic view is the view that has dominated our way of thinking since the seventeenth century. The way we think the world and the nature works is quite important since it reflects on the way we look upon organizations.

This chapter will start with a description of the origin of the mechanistic view. Then we will see how this way of thinking can be applied on organizations. Next a description of the management style that stems from the mechanistic view, command-and-control, will be presented. In the last section, Software Process Improvement will be compared against the mechanistic view.

Origin

The mechanistic view doesn't have a clear path of evolution—instead there are many various people and groups that have contributed in different ways. It has its origin in the fifth and sixth century B.C. (Sanders, 1998). The atomists thought that everything in the material world consisted of small lifeless particles, that is atoms. They believed that the world, and even life itself, was a well-constructed machine. The Pythogoreans considered the world as primarily numerical, where numbers were more important than matters. Plato considered the world as being constructed by triangles. This made both the Pythogoreans and Plato believe that mathematics was the primary tool to understand nature.

Aristotle, on the other hand, had a more qualitative worldview than the quantitative worldview of his teacher Plato, and the Pythogoreans. In his worldview change is due to the potential of change residing in all objects. All objects change to become what they are intended to. This view of change differs quite much from the proponents of the mechanistic view, where change is due to our inability to perceive the underlying stability.

The worldview of Aristotle, which was a more organic worldview, dominated the way of thinking for almost two thousand years.

During the Scientific Revolution of the seventeenth century the quantitative worldview was revived. In the end of the seventeenth century Aristotle's natural philosophy had been replaced by a mechanical philosophy. According to Sanders (1998) the mechanical philosophy was a reaction against attributions of humanlike qualities to nature, since that view provided little aid to understand nature. Instead, considering the world as being a machine was considered a more powerful tool.

Mechanistic Organization

Much more could be said about the mechanistic worldview and its consequences. However, now we are going to concentrate on the implications of using machines as metaphors for organizations. As Morgan (1986) states: “[f]or the use of metaphor implies *a way of thinking* and *a way of seeing* that pervade how we understand our world generally.” (p. 12) So by studying the machine metaphor in an organizational context we also learn more about the mechanistic worldview.

According to Gharajedaghi (1999) the mechanistic view provided the basis for the Industrial Revolution. With the Industrial Revolution the concept of organizations was created using the metaphor of a complicated tractor where all the parts perform simple tasks. Since the concept of organizations originated from a mechanistic view our traditional view of an organization is a mechanistic view.

But what are the consequences if you use a machine as a metaphor for an organization? The essence of the metaphor is that the organization has no purpose of its own, it’s a mindless system (Gharajedaghi, 1999). For the organization to be useful it must fulfil the need of the owner for which it is designed. Organizations are also expected to show the same characteristics as a machine in terms of predictability and repeatability. “People are frequently expected to arrive at work at a given time, perform a predetermined set of activities, rest at appointed hours, then resume their tasks until work is over.” (Morgan, 1986, p. 20)

Just to see how an example of how influential metaphors are, and how dominating the machine metaphor is, it could be interesting to look at the first paragraph of the Background section of the Introduction chapter. In this paragraph I am talking about the problems we had in a project with getting the right structure for the group. This description shows clear traces of a mechanistic worldview since the structure, or the organization, is regarded as a tool and has no value on its own—the organization is described as a way to solve a specific problem.

So if we look upon an organization as a tractor it should provide a good description of how mechanistic organizations ought to work. The tractor exists to serve the farmer. For the farmer to trust his tractor it must be reliable—it can’t have a mind of its own. For this to be possible all parts of the tractor must be reliable—the wheels can’t decide by themselves that they want to take a left turn. All parts must follow the orders given from the central authority, the farmer. Moreover, it must be predictable—the tractor must behave in a consistent way so the farmer knows what outcome he can expect given a certain input. There is also a clear cause and effect characteristic. When the farmer turns the steering wheel it has the effect that some inner parts turn and move. These parts, after a series of cause and effect steps, make the wheels turn. Turning the steering wheel does not sometimes make the engine to increase the speed. All actions follow a pre-defined, linear route.

All the parts in the tractor are specialized to fulfill their simple tasks. In itself a wheel is not very useful. But together with the other parts needed for a tractor the whole is quite useful. Because of the specialization of all the parts it’s easy to replace a broken part as long as you manage to find a corresponding part. Another consequence of the specialization is that it is quite hard for a part to do something outside its predefined task—a front wheel can’t easily switch places with the steering wheel.

The tractor cannot adapt itself to changes in the environment. If the farmer wants to use the tractor in an environment for which it wasn’t originally designed he must reconstruct it. Furthermore, it is important that the tractor works efficiently. If it doesn’t work satisfactorily the farmer tries to optimize the different parts so that the whole tractor works in an efficiently manner.

Talking about mechanistic organizations might seem a little diffuse. But if we instead look at another, more familiar, concept—bureaucratic organizations—it becomes more concrete. Bureaucracies, which are practically mechanistic, have (in Max Weber’s model) six elements (Youngblood, 1997, p. 21):

- A clear-cut division of labor
- A hierarchy of authority
- Recruitment of managers based on technical knowledge and expertise
- An explicit set of rules for making decisions
- A strict separation of business and personal concerns
- The establishment of career employment

Even though the word “bureaucracy” has become less popular, 95% of all companies are still bureaucracies, according to Youngblood (1997, p. 30). A bureaucratic organization is designed for stability and to produce predictable and certain results. It is well suited for stable environments. However, the concept evolved in a time where the problem was to control a large mass of unskilled workers (former farmers).

Command-and-Control

Mechanical parts differ to some extent from humans. A mechanical part works practically in the same way from the moment it is constructed until it breaks. To make employees imitate this behavior, management is a key issue in a mechanistic organization. The management style used in mechanistic organization is often referred to as command-and-control.

The central authority in the organization makes the decisions and commands the subordinates to carry out his orders—hence it leads to compelled behavior. However, humans are usually not as reliable as mechanical parts. To handle this the subordinates must be controlled so that they work in accordance with what they are supposed to.

With time and growth of the organization giving direct orders about everything becomes harder. One way to manage without some of the orders is to replace them with rules, regulations, and predefined processes. However, even if the need of active command from the central authority decreases, there might be an increased need for monitoring the subordinate to ensure that they work in accordance with the rules and orders.

With increasing size of the organization the central authority loses its initial effectiveness—there is a need for decentralization. But according to Gharajedaghi (1999) decentralization is inconsistent with the principles of no deviation and unity of command—the different authorities within the organization won’t make a coherent command. This can be compared with the tractor—different wheels will turn in different directions. Gharajedaghi states that this is not ideal in an organization and that it will lead to chaos and suboptimization. He also thinks that this might be the reason why most large organizations oscillate between centralization and decentralization.

In command-and-control the chain of command runs from the top to the bottom (Morgan, 1986). Each employee should have at most one superior from which he or she gets the orders from above. This implies that there, from any position in the bottom, is only one route to the top. These routes make up the predefined channels for orders, as well as communication.

I doubt that there is anyone, at least in the software literature, who advocates command-and-control. However, I do believe that there are clear traces of command-and-control both in practice and in literature. When the software literature talks about leaders, managers, or management, they are mostly concerned with the appointed leaders. If we look at IEEE standards definition of software project management:

> "Software project management is the process of planning, organizing, staffing, monitoring, controlling, and leading a software project" (Bennatan, 1994, p. 39)

we realize that it indicates traces of command-and-control. If we look at Gilb's (1988) description of a manager's tasks:

> "The tasks of the manager are to define objectives, to create, evaluate, and select alternatives for reaching them, and to control the implementation of selected alternatives." (p. 15)

it shows a strong relation to command-and-control where the manager makes the decisions and make sure that the subordinates fulfil the decisions.

According to Humphrey (1997) it is important for the manager to get his or her will. "Employee commitment can be obtained only by leadership. Leaders must establish the goals and convince their subordinates to accept them as their own." (p. vii) It is also important that the managers are persistent about their decisions and orders. "Once leaders make decisions, they must drive to overcome all obstacles, and when their people are ready to give up, they must rally them for another try." (pp. 4-5) Moreover, "[m]anagers must strive to get the job done, and when their groups do not perform, they must step in and straighten things out." (p. 169) And when the organization has serious problems with for instance attendance problems, "the first-level management is typically not properly supervising their people and keeping them busy." (p. 226)

Humphrey (1997) also highlights the need for decentralization of the authority when the organization get large. "Beyond a few dozen professionals, a controlling management style is no longer practical . . . everyone has some limit beyond which he or she can no longer personally make all the decisions." (p. 227) So when the size grows, the manager must have a management team to which he or she can delegate authority and responsibility. This implies that even though there are some decentralization, the authority has just been spread out to a larger number of authorities. For the senior managers to direct the work they must convince their subordinates, the first-line managers. "When senior managers do convince them, however, subordinate managers can properly direct their people, track their progress, and take timely corrective action when things go wrong." (p. 230)

Strengths and Weaknesses of Machine Metaphor

Strengths

The machine metaphor is very popular and seems to be quite appealing to many people (Morgan, 1986). Mechanistic approaches are very good at achieving high efficiency of certain kinds of work tasks. They are also a way to achieve and maintain a close control over people.

When Morgan is describing the strengths of the machine metaphor, he looks at the conditions for a mechanistic organization:

- the task is straightforward
- the environment is stable enough
- you want exactly the same product every time
- the precision is at premium
- the people are compliant and behave as designed

Moreover, he is pointing out organizations where a mechanistic approach has been successfully applied, such as the McDonald's hamburger chain. The McDonald's franchise outlets are all able to produce a uniform product that is both predictable and repeatable—no matter where you go in the world you still know what to expect. They manage to achieve this by using mostly students following centrally defined specifications.

Weaknesses

As with all metaphors, the machine metaphor is highlighting some aspects and de-emphasize others. By equating and reducing people working in organizations to mere mechanical, lifeless, parts the machine metaphor is "forcing [the organizations'] human qualities into a background role." (Morgan, 1986, p. 13) Morgan is also stating that although the machine metaphor has sometimes contributed to increases of the productivity, these have been achieved at great human cost. "Employees lose opportunities for personal growth, often spending many hours a day on work they neither value nor enjoy, while organizations lose the creative and intelligent contributions that most employees are capable of making, given the right opportunities." (p. 38)

There are also other weaknesses with the machine metaphor according to Morgan (1986). Since mechanistic organizations needs a stable environment to be able to achieve their predefined goals, they might not be able to adapt when the conditions are changing. So when an unfamiliar situation occurs, for which there are no predefined actions, it might be ignored or tackled using existing processes or policies. This makes them quite poor at handling changes—they are designed for repeatability, stability and precision, not innovation. Mechanistic organizations might also lead to a bureaucracy where nothing is questioned and the people are not thinking by themselves. People are there to do what they are told and what is written in their job description. This behavior is encouraged by removing the controlling and monitoring functions from the front-line employees, and thus remove their responsibility of their work. Another drawback with the machine metaphor is if people are driven more by their own interest than by the goals of the organizations they might not perform the work intended. Such personal interests might be than the persons are more eager to acquire more power and making a career in the hierarchy by competing with others, than to look at what is best for the whole organization.

Software organizations, at least a large part of them, live in a complex and chaotic environment, as described in the introduction chapter. Olson (1993) argues that several software development activities, such as problem-solving, requirements gathering, design and forecasting, are chaotic. At the beginning of a problem-solving effort there is only the problem. But it is often not enough with the first solution, and the problem is not the same any more because the new problem consists of the original problem and the first solution. As time passes more solutions are applied, the environment changes, and what once was a simple problem creates a chaotic and complex weave with its own problems, where it is hard to distinguish the problems from the solutions. "Even if the initial solution is officially abandoned, its effects will be felt in most subsequent solutions, a potentially undesirable legacy." (p. 56) In the beginning of the requirements gathering process the user do not usually know what is needed and how to turn it into requirements. To identify the actual requirements the users and the developers have to engage in interaction or several cycles of interaction. With help of this feedback process the users form mental models of what the developers have to work with, and the developers form mental models of what the users want. Then the users and developers can evaluate each other's mental models, which might not fit together, and try to align them. It usually takes several cycles, depending on the interaction between users and developers, before the requirements definition reaches an acceptable level. The mental models change over time, change as the developers and users are interacting, and depend highly on the result of previous cycles. Together these different levels of feedback make requirements gathering a chaotic activity, which seems to come quite close to problem-solving. The design activity does not have to be chaotic if the problem is well known and built on a well-known base, according to Olson. But when the problem is not well known and the base the problem is to be built on has to be created, a chaotic approach is more appropriate. In the chaotic approach you start by quickly designing solutions to parts of the problem, then you review the design, code prototypes, which you present to a user for feedback, and finally compose a product prototype iteratively from successive functional prototypes until you understand the problem. Then you can gather the different pieces and make them work together. Since design is a kind of problem-solving activity it is not unreasonable to think that design is chaotic as well as problem-solving in general.

Software Process Improvement

It is quite safe to assume that there are no organizations that works entirely according to the machine metaphor. However, the attempts to make organizations predictable, reliable, independent of individual employees and the other mechanistic characteristics are quite clear. One movement in the software industry that aims at these mechanistic goals is Software Process Improvement. Software Process Improvement, and the related movement Software Engineering, have been used to deal with the "software crisis". Software crisis is an expression used to describe the delivery and quality problems of the software industry and was especially used to describe the state of the industry of the 1960's and 1970's. There are probably different opinions, but many people believe that Software Engineering and Software Process Improvement have more or less "cured" the software crisis.

Software Process Improvement has its roots in the manufacturing industry (Zahran, 1998). The main influences were Total Quality Management (TQM) and different statistical quality control methods. In fact, one of the main Software Process Improvement initiatives, the Capability Maturity Model (CMM), is an application of TQM to software (Paulk et al, 1995). Since the manufacturing industry is traditionally built on mechanistic principles the chance, that Software Process Improvement shares these principles, is quite large.

There are some clear similarities between Software Process Improvement and the mechanistic view, the dominating metaphor seems to be the machine metaphor. One is that a mature process—the kind of process that is aimed for—is a disciplined process, as stated by Zahran (1998). The word discipline has many meanings, but Zahran has chosen the meaning: "the system of rules used to maintain control or order over a group of people" (p. 17). Thus, when you have attained a disciplined, or mature, process you have control. It is not hard to see the resemblance with the command-and-control management style.

In order to achieve an effective process Zahran mentions three important components: definition, training and mechanisms to ensure that the defined process is followed.

Zahran states that if the process is not defined and documented enough a number of risks arise, as described below. The first risk is the following scenario: “when a new team *leader is too weak to enforce* the process with the same vigour as his or her predecessor” (p. 20, my emphasis). This quote shows both the importance of the central authority and its ability to command the employees, just as in the mechanistic view and the command-and-control management style. The other risks Zahran mentions are that gurus or disruptive persons join the team, or that the process is interpreted differently among the team members. Such differences among the people involved are harmful for the discipline and should be avoided. To handle these problems process definition is, according to Zahran, a key issue.

It is not sufficient to define and document the software process. The employees must be trained to make use of the process definition. The training in conjunction with the definition represent the command part of the command-and-control management style—the central authority commands the employees to work according to defined process. The word command is perhaps a bit too extreme. It is usually possible for the employees to give feedback on the defined process. However, it’s still the central authority that is in command.

The third component, of what makes an effective process, is enforcement. If there is no enforcement the employees will be tempted to avoid following the process (Zahran, 1998). This enforcement component corresponds to the control part of the command-and-control management style.

Summary

- The way we think the world and the nature work reflects on how we think organizations work, or how they ought to work.
- The mechanistic view is the view that has dominated our way of thinking since the Scientific Revolution.
- In the mechanistic view an organization shall have a central authority in order to attain unity of command.
- It is important that a mechanistic organization is predictable and reliable with a repeatable behavior.
- The parts of a mechanistic organization are all specialized and do only need to handle simple actions. All the various simple tasks together create the complicated behavior needed for the organization.
- Any action in one part of the organization cause an effect in another part, in a linear manner, following a predefined route.
- Bureaucratic organizations are one kind of mechanistic organizations, and constitute 95% of all organizations.
- The management style that stems from the mechanistic view is command-and-control.
- One main strength of the machine metaphor is that, when the circumstances are right, it can lead to organizations that work in a efficient, repeatable and predictable manner producing exactly what they are designed to.
- The main weaknesses of the machine metaphor are both that it doesn’t put any focus on the human qualities of organizations, and that the organizations tend to be quite unable to be innovative or react to changes.

- Most software organizations live in an ever-changing, chaotic, environment and some of the most important activities in software development are chaotic, which implies that a mechanistic approach might not be optimal.
- It is easy to find support for a command-and-control management style in the software literature.
- Software Process Improvement has its roots in the manufacturing industry and shares at least some of its mechanistic view

---

The mechanistic view has been proven effective in the past. It influenced the Industrial Revolution, which decreased the massive unemployment and achieved mass-production. In the software industry it has been proven successful as well—it helped reducing, or even curing, the software crisis. But is it still a better view to build organizations in general, and software organizations in particular, upon or are there any other?

CHAPTER 3

THE ORGANIC VIEW

This chapter will describe the science of chaos and of complexity, which at times is called the *new science*. As stated by Youngblood (1997), modern scientists have for long known that the world doesn't work as a machine. "Nature is very much alive and actively organizing itself into ever-higher levels of complexity and order." (p. 6) However, society has not left the mechanistic worldview. It still influences the way we think.

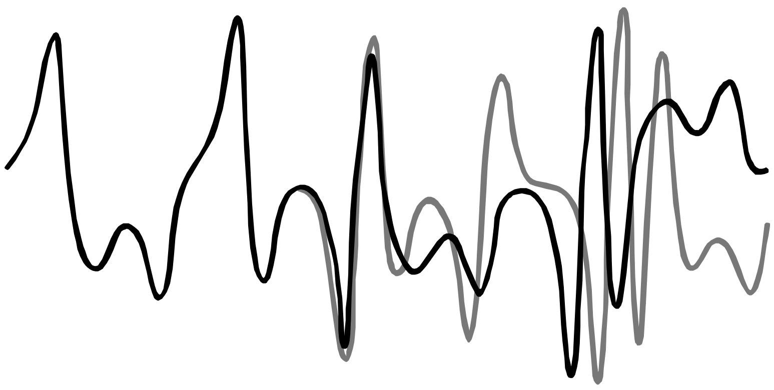
This chapter will start by a description of the chaos theory. Then a description of complexity theory, and its relation to chaos theory will follow. Chaos and complexity provides the foundation for the organic view. Based on chaos and complexity theory a description will be provided of what order is according to the organic view. Some aspects of organizations and leadership will then follow. In the last section some more organic approaches to software development will be presented.

Chaos

The study of chaos has its origin in Edward Lorenz research in weather prediction in 1960 (Gleick, 1988). Since Lorenz had a background in mathematics he believed in using mathematics in order to predict changes in the weather. In an attempt to mimic the behavior of the weather Lorenz created a computer model, using twelve equations. Since the model was based on equations it was deterministic, thus produced the same result given the same preconditions. He managed to create a model that he thought conformed to the way he perceived the weather—over time you could distinguish some patterns, however there were always irregularities—it was an orderly disorder.

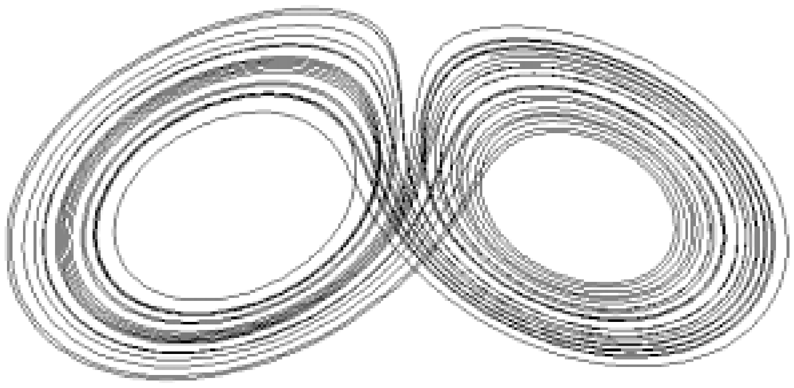
One day Lorenz ran a simulation twice. When he compared the printouts he noticed that they were very divergent. The first months were quite similar but after a while all resemblance was gone. This was quite puzzling since they were supposed to look the same—the program had not changed and he had entered the input himself. Eventually he noticed that the result was due to an error in the input, the input to the first simulation was .506127, whereas it was .506 for the latter. What he had discovered is called "sensitive dependence on initial conditions", which is more commonly known as the Butterfly Effect (see Figure 3.1). The Butterfly Effect is a metaphor for a butterfly flapping its wings in Asia resulting in a hurricane in the Atlantic.

Figure 3.1 The Butterfly Effect



The discovery of the Butterfly Effect—that small changes cascade upward through a system resulting in large consequences—made Lorenz realize that long-term predictions of weather are impossible. If the Butterfly Effect wouldn't exist the weather would—as soon as it reached a state that was close to a previous state—repeat itself periodically. In fact, "[Lorenz] realized that *any* physical system that behaved nonperiodically would be unpredictable" (Gleick, 1988, p. 18). Lorenz continued to work with aperiodic, nonlinear systems, that is systems that don't repeat themselves, and managed to create such a system called the Lorenz attractor (see Figure 3.2). It's an aperiodic system that shows the apparent order that appears in the disorder, using only three equations.

Figure 3.2 Lorenz Attractor



At the time when Lorenz discovered the Butterfly Effect science was very specialized—each scientist concentrated on narrowly focused research questions within their discipline (Sanders, 1998). This was natural in the mechanistic view, since the only way to understand the whole was to understand the parts. This way of working is not suitable in nonlinear systems, such as the Lorenz attractor, since the variables are not independent as in linear systems. The variables depend on each other—they are interdependent. This has the effect that you have to use a more holistic and interdisciplinary approach.

Moreover, for the scientist to be able to model different systems, irregularities were disregarded. The assumption was that "[g]iven an *approximate* knowledge of a system's initial conditions and an understanding of natural law, one can calculate the *approximate* behavior of the system" (Gleick, 1988, p. 15). This kind of reasoning doesn't apply on nonlinear systems due to the Butterfly Effect.

As more and more scientists started looking into chaos theory, they realized that chaos seems to be everywhere. In fact, "*[m]ost of the world is made up of nonlinear systems. The world is more nonlinear than it is linear*" (Sanders, 1998, p. 66).

Complexity

When the scientists started to understand the nonlinear systems they began looking on the interactions within and between systems (Sanders, 1998). The systems that they started to study are called complex adaptive systems (cas). Thus, chaos theory is concerned with nonlinear systems, whereas complexity theory is concerned with complex adaptive systems. The complex adaptive systems are a special type of nonlinear systems (Youngblood, 1997). Nonlinear systems can be snowflakes and chemical reactions, whereas complex adaptive systems are nonlinear system “with the added ability to use its behavior and actions to actively sustain an identity” (p. 34).

Complex adaptive systems can be found everywhere and assume the most diverse appearances. Some examples of complex adaptive systems are: ecosystems, the human immune system, economies, and central nervous systems (Holland, 1995). A complex adaptive system consists of a collection of agents, each following their own set of principles when interacting with the other agents. Since the systems can be so diverse, the agents can be very diverse as well. The agents in an ecosystem are the different species—in the human immune system they are antibodies.

A complex adaptive system doesn’t remain the same for too long. It is continuously adapting. The stimuli for the adaptation come from the environment and the agents’ gained experience (Holland, 1995). The time it takes for a system to adapt, or learn, can be as short as seconds to hours for a central nervous system, and as long as years to millennia for an ecosystem.

All complex adaptive systems share the same general principles that affect their behavior. According to Holland (1995) there are seven common characteristics, or basics:

- Aggregation
- Tagging
- Nonlinearity
- Flows
- Diversity
- Internal models
- Building blocks

Aggregation

The different agents might have a rather simple behavior by themselves. The complex behavior a system can achieve is due to the aggregate interaction of the agents. Holland uses the example of ants. An individual ant has a stereotyped behavior and cannot adjust to changes in the environment that requires a different behavior. In contrast, an ant nest is very adaptive and survives large changes in the environment. An aggregate of agents can also work as an agent, a meta-agent. This meta-agent collaborates with other agents and meta-agents. Hence, they aggregate into another meta-agent with an even more complex behavior than the aggregate agents. These aggregates, and aggregates of aggregates, create a hierarchical structure.

Tagging

But for a complex adaptive system to emerge it isn’t sufficient to place a couple of agents together. The mechanism that facilitates the creation of aggregates in complex adaptive systems is called tagging. An example he mentions is a banner or a flag used to rally members of an army. Another function of tags, besides creation of aggregates, is to let the agents see with which agents they can interact.

Nonlinearity

To understand the dynamics of a complex adaptive system it isn’t sufficient to understand the actions of the different parts, for instance you don’t understand the dynamics of an ecosystem by just having a catalog of the activities of most of the species. This is due to the nonlinearity of complex adaptive systems—the whole is more than, or at least different from, the sum of the parts. However, if a complex adaptive system would have been linear, as in the mechanistic view, knowledge of the action of its parts would have been sufficient in order to understand the system as a whole. In linear systems, action is more important than interaction, whereas interaction is more important than action in nonlinear systems. Since complex adaptive systems are nonlinear they have no clear cause and effect characteristics.

Flows

For flows to be possible in a complex adaptive system there are nodes, connectors, and resources. Nodes are agents; connectors are the possible interactions (where tagging plays a crucial part); resources are whatever is exchanged between the agents. In an ecosystem the species would be the nodes, the connectors would be the food web interactions; and the resource would be biochemicals. According to Holland there are two important properties of flows in all complex adaptive systems—multiplier effect and recycle effect. The multiplier effect takes place when resources are inserted into a system at an agent. When an agent receives resources it chooses if it should transform them. Then it passes the resources to other agents, thus creating a chain of changes. Holland uses the example of a contract to build a new house, which causes a chain of reactions with a five-fold increase of the flow of resources through the economic network. The recycling effect is due to cycles in the networks and has the effect that resources can remain longer within the system.

Diversity

Diversity is very important for the survival and prosperity of a complex adaptive system. There is a need for a large diversity of agents, all filling different niches. However, the diversity is not created by random—the different niches that are needed in a system are defined by the interactions of the agents. The different niches needed get filled through a phenomenon called convergence, which occurs when an agent that fills a certain niche leaves the system. This creates a hole that needs to be filled, for instance through another agent taking the place. The diversity pattern is not constant—it evolves just as everything else. One reason for evolution of the diversity pattern is when new agents, with new interaction possibilities, join the system.

Internal models

The mechanism of internal models is used for anticipation. Since the variety of the input is so large, details are eliminated in order to see the underlying pattern. The agents must then convert the selected input patterns to changes in their internal models. These changes must enable the agent to anticipate what will happen when the pattern reappear. Internal models can be either tacit or overt. Tacit models are models that implicitly affect the way an agent anticipates and acts, whereas overt models are explicit explorations of alternatives.

Building blocks

For an agent to be able to anticipate when exposed to a new situation the mechanism of building blocks is used. Through decomposing the environment into building blocks—for which the agent already have internal models to anticipate—the agent can combine them to a more complex behavior to handle the new situation. The creation of new internal models using building blocks is much faster when the internal models are overt.

Youngblood (1997) has also a model for complex adaptive systems, or self-organizing systems as he chooses to call them. This will be provided below as a complement to Holland's model. This model has similarities with Holland's model but do also add some different aspects. In the model there are seven principles:

- Wholeness
- Connectedness
- Identity
- Balance

- Creativity
- Openness
- Flexibility

Wholeness Complex adaptive systems have emergent properties, which implies that the whole becomes more than the sum of the parts. You cannot reduce the system down to its components without losing the uniqueness of the system. As said before, you can't expect to understand the dynamics of an ecosystem just by knowing the activities of the species. "[T]he whole organizes the parts, and any sense of order at one level derives from the self-organizing activities at a higher level." (p. 41) These emergent properties have the nature that they are impossible to build since they must emerge. What you can do is to provide circumstances for them to develop gradually over time. This does not accord well with the mechanistic view where you try to optimize and re-engineer the organization of a system in order for it to be as an efficient tool as possible.

Connectedness The different parts of a complex adaptive system are interrelated in nonlinear, complex and intricate ways. This makes it very hard to realize any clear cause and effect patterns (think of the Butterfly Effect). The connectedness is also influenced by the concept of aggregation (as described in Holland's model). However, the connectedness is not only concerned with only one system. Complex adaptive systems are not able to thrive, or even survive, without a high degree of connectedness with its environment. "These symbiotic interactions are so tightly entwined that systems cannot be considered as separate from their environments." (p. 43) This symbiosis makes it crucial for the system to consider the well-being of its environment. "The qualities of one will dictate the qualities of the other." (p. 44)

Identity The identity of a complex adaptive system is the shared sense of purpose of the system. It is a central concept that guides the action of the different agents to create a coherent whole. The identity plays an integral part for the relation between integration and self-assertion. In complex adaptive systems there is a continuous interplay between integration and self-assertion. Integration occurs when the parts are cooperating and acting as being part of the whole, which makes integration crucial for the system in order to minimize sub-optimization. Self-assertion, which is practically the opposite of integration, is an aspiration for autonomy and is essential to attain diversity of the agents. Both integration and self-assertion are needed for the system, "the agents are both cooperating in sustaining the whole and expressing their unique identity through autonomous action." (p. 45) To really clarify the distinction, Youngblood states that what is aimed for is individuation and not individualism. This implies that the agents are highly individual, but at the same time show a high accountability for the well-being of the system. Individualism would have implied that the agents had only been concerned with maximizing their own well-being. It is the identity of the system that helps the agents balancing between the welfare of self and the system as a whole.

Balance The balance of complex adaptive systems is not a static state, but rather a dynamic, fluctuating state. The fluctuations might be rather large, but keep within tolerable limits. This balance must be attained within the system as well as between the system and its environment. A balanced system does not have an optimal state where all the parameters are perfectly adjusted, since that would mean that it was static. The most beneficial state for the system is when all the parameters are fluctuating within tolerable limits. There is a risk trying to maximize a parameter since it is bound to disrupt the balance of the whole system and thus being counterproductive, according to Youngblood.

Creativity In the mechanistic view such things as reliability, predictability and repeatability are considered essential and variations and disturbances are not beneficial. In order to achieve this in a system there is a need for controlling activities, which are called negative feedback. Negative feedback is used to reduce variation and facilitate self-renewal. In the mechanistic view, creativity is new ways to increase the negative feedback—new ways to make sure that the system acts in a more predictable manner and to minimize the disturbance. In contrast, the creativity in complex adaptive systems is called self-transcendence and uses positive feedback. The positive feedback amplifies small disturbances until they reach a critical mass. This makes the system reach new, higher, levels of complexity and organization. For this self-transcendence to take place there has to be a certain amount of "messiness", or redundancy. The long-range effects of redundancy can, through positive feedback, enable the system to perform brand new actions.

The effect of positive feedback becomes even more influential when there are reinforcing feedback loops (called recycling effects in Holland's model). "In a reinforcing loop, behaviors continue to escalate until the system hits natural limitations or transcends itself to achieve a wholly new and more stable structure." (p. 55)

Openness

There is a need for three different parameters for self-transcendence to take place, namely information, diversity and interactions. Openness towards all of these parameters is beneficial for the system. It should be open for the agents to exchange as much information as possible, to anyone of the other agents, and for them to become highly diverse. Greater openness implies greater potential for creativity. Youngblood uses an analogy to describe information, diversity and interactions: "[i]f information is the lifeblood of systems, then high-quality, diverse interactions are the beating heart." (p.64)

Flexibility

In complex adaptive systems, the substance is more significant than the form. It is not the form that holds the system together, but rather the substance. The form and structure emerge and dissolve as needed to fit into the current situation. The substance and situation defines the structure and not the other way around. This makes complex adaptive systems very flexible. They also have to cope with energy and matter leaving the system. To do so they use the flow mechanism (as described in Holland's model). Another function of flow is that it helps the system being flexible towards changes in the environment. If there isn't a healthy flow—both inflow and outflow—in a complex adaptive system, it will inevitably lead to death of the system.

Order

The order in the organic view is not the same as the order in the mechanistic view. The order in the organic view emerges from disorder, whereas the order in the mechanistic view is an imposed order that stems from a centralized authority. The order in the organic view is not characterized by quiet or calm, but rather a self-organizing pattern (Sanders, 1998).

The place where the order emerges in complex adaptive systems is the area between chaos and order—the edge of chaos (Youngblood, 1997). When the systems have too little structure—are too close to chaos—they get too sensitive for disturbance. When they have too much structure—are too close to a static order—they get too rigid to cope with changes. Hence, for a complex adaptive system to stay well it's important that it's balancing at the edge of chaos. The effect this balancing act has is, according to Youngblood, to make the system self-organize, thus creating a higher level of order than an imposed order—an order that is more creative and stable than an imposed order.

In the organic view rigor is not a goal, as opposed to the mechanistic view. Instead rigor is used as a means of balancing the system at the edge of chaos

(Highsmith, 1999). This means that instead of applying as much rigor as possible, just enough rigor is applied to prevent chaos.

A complex adaptive system is constantly exposed to a flow of information, which cause changes to the system (Sanders, 1998). Most of the changes are minor, but some of them cause the system to lose its balance. This makes the system to go through a period of instability, or chaos. However, after a period of instability the system adapts and self-organize to cope with the new conditions and regains its balance. To prevent that the period of instability—when the form of the system dissolves—leads to death instead of self-organization it's important, according to Wheatley (1999), for the system to maintain its identity.

Organizations

In the organic view the metaphor for an organization is complex adaptive systems. Highsmith (2000) goes a little further and states that all organizations are, in fact, complex adaptive systems. This means that what is applicable on complex adaptive systems should be applicable on organizations as well.

Hock (1999) struggled with three questions: "Why are organizations, everywhere, whether political, commercial, or social, increasingly unable to manage their affairs? Why are individuals, everywhere, increasingly in conflict with and alienated from the organizations of which they are part? Why are society and the biosphere increasingly in disarray?" (p. 2). As he struggled with these questions he became the founder and CEO of VISA. When he retired he discovered complexity science. What amazed him were not the theories themselves, but rather that science had derived the same concepts regarding organizations from studying biological systems as he had used for a long time. However, since he thought that there was one word missing, he created his own from the words chaos and order.

> *chaord* "1. any self-organizing, self-governing, adaptive, nonlinear, complex organism, organization, community, or system, whether physical, biological, or social, the behavior of which harmoniously combines characteristics of both chaos and order 2. an entity whose behavior exhibits observable patterns and probabilities not governed or explained by its constituent parts 3. any chaotically ordered complex 4. an entity characterized by the fundamental organizing principle of evolution and nature" (p. 30)

This discovery led Hock to return to organizations to turn them into chaordic organizations. When designing a chaordic organization there are six elements that are searched for:

- Purpose: the statement of purpose that binds the organization together. It is important that the purpose is understood and shared by everyone and that is something meaningful.
- Principles: the principles are descriptions of how the whole and the parts shall conduct themselves in the pursuit of the purpose, and they are often both moral and ethical.
- People: when the purpose and the principles are agreed upon it's time to identify the parties needed in order to achieve the purpose in accordance with the principles.
- Concept: the organizational concept that is needed.
- Structure: the structure is the embodiment of the purpose, principles and concept into a constitution with details such as ownership and voting.

- Practice: the practice is the deliberations, decisions and acts of the involved parties that evolve.

This process of forming an organization indicates a fundamental difference between the organic and the mechanistic view. If you look upon organizations as machines they don't have any purpose of their own. Using the metaphor of a tractor as an organization, if the tractor can't be used by the farmer to make profit and to facilitate the farmer's work it is useless. "Making a profit is not a purpose. It may be an objective; it may be a necessity; it may be a gratification; but it is not a purpose!" (p. 8). Making profit isn't something meaningful, something that is worthy of pursuit.

In Youngblood's (1997) model, which he calls the Organizational EcoSystem Model, there are seven internal aspects of an organization: purpose, principles, strategy, culture, structure, process, and technology. This model also emphasize purpose and principles. Youngblood states that it's crucial that the organization manages to create a shared sense of purpose and principles, which leads to alignment of the different parts. If the parts aren't aligned it will lead, according to Youngblood, to self-imposed anarchy. This, in turn, will justify a need of command-and-control management style. As Youngblood states, alignment is the difference between laser and normal light, and between a magnet and a piece of iron. To align the parts in an organization through purpose and guiding principles is one of the most important roles for the management in Quantum Organizations (which is Youngblood's term for organizations based on the science of complex adaptive systems).

Purpose and principles are the core ideology, the culture, of an organization (Youngblood, 1997). Where mechanistic organizations try to achieve alignment through command-and-control, organizations based on an organic view must rely on purpose and principles for alignment. When everything else changes the purpose and principles remain the same. It is the identity of the organization, which helps preventing the organization from dissolving when going through periods of instability. However, sometimes it's important to change even the core ideology.

Leadership

To manage a system, such as an organization, is not so much about control (Gharajedaghi, 1999). What is left, when you realize that you can't rely on control, is leadership. "*Leadership* is . . . defined as the ability to influence those whom we do not control." (p. 32) Where the management style in the mechanistic view is command-and-control, the management style in the organic view is leadership-and-collaboration—leadership replaces command, and collaboration replaces control (Highsmith, 2000).

Weinberg (1986) takes an organic problem-solving approach to leadership, although not explicitly based on complexity theory. However, his approach has many similarities with complexity theory. Weinberg's definition of leadership is:

> "*Leadership* is the process of creating an environment in which people become empowered" (p. 12)

This approach is apparently different from command-and-control. The traditional approach divides people into those who organize and those who get organized—those who are destined to lead and those who are destined to follow. Weinberg makes no such division, everyone contains the ingredients for leadership. What differs between people is how well the ingredients are developed.

"[M]ost everyone likes to organize some of the time, and perhaps be organized some of the time, depending on the situation." (p. 204) This implies that there cannot be just a few appointed leaders but instead everyone leads. So, instead of concentrating on who should lead, Weinberg suggests we should concentrate on getting the job done.

Hock (1999) has a similar view of leadership. He states that:

> "A true leader cannot be bound to lead. A true follower cannot be bound to follow" (p. 67)

The reason behind this is that the relationship leader/follower requires freedom to choose. If this relationship isn't chosen voluntarily, it will lead to compelled behavior, as opposed to induced behavior. "Where behavior is compelled, there lies tyranny, however benign." (p. 67) Hock's view of leadership differs quite much from the traditional view. He describes that he often asks people what the most important responsibility is for a manager. The answers are often diverse, such as selecting the employees, motivating them, training them, organizing them, and controlling them. These answers all share the same property that they are downward-looking. That makes leadership something for the few.

However this is not how Hock views true leadership. Instead, "[t]he first and paramount responsibility of anyone who purports to manage is to manage self; one's own integrity, character, ethics, knowledge, wisdom, temperament, words, and acts." (p. 69) If a person can't manage self, nobody will choose to follow. "The second responsibility is to manage those who have authority over us: bosses, supervisors, directors, regulators, ad infinitum." (p. 69) If you don't have their support you will be very limited. "The third responsibility is to manage one's peers . . . associates, competitors, suppliers, customers—the entire environment, if you will." (p. 69) Without their support you will be very limited, as well. The fourth, and the last, responsibility is "to manage those over whom we have authority." (p. 70) According to Hock, half of the time should be spent of managing self, a quarter of the time on managing superiors, a fifth of the time on managing peers. This leaves almost no time to manage subordinates, the ones over whom we have authority. There are no need to spend too much time and effort on managing subordinates, as long as they manage themselves, their superiors, their peers, and their subordinates. Since there are no way to use a command-and-control management style on self, superiors, and peers you only have leadership left to use. Hence, everyone needs to be a leader, just as Weinberg suggests.

Strengths and Weaknesses of Organic Metaphor

Although Morgan (1986) doesn't deal with any metaphor that exactly corresponds to complex adaptive systems, he discusses an organismic metaphor, that is organizations as organisms. Organisms are not the same as complex adaptive systems, but rather a subset of complex adaptive systems. Most of the strengths and limitations of the organismic metaphor Morgan identifies are also applicable on the metaphor in this thesis.

Strengths

One strength is that the metaphor emphasizes understanding of the relations between organizations and their environments. This is something that the machine metaphor more or less fails to take into account. Instead of being a closed system it is an open system with a constant exchange with the environment where the processes are more important than the parts.

Since survival is the key aim in this metaphor, much attention is given to fulfil the needs of the organizations. According to Morgan this will improve the management of organizations since survival is a process and not a target or an end point as goals are. This makes the management more flexible and the goals are not ends in themselves, but rather secondary to the process of survival. Another consequence of the emphasis on the needs of the organizations is that it “encourages us to see organizations as interacting processes that have to be balanced internally as well as in relation to the environment.” (p. 72)

Another strength is that it put a focus on the effect of organic organizational forms in the process of innovation. This is a great advantage over the machine metaphor if innovation is a priority, since then organic organizational forms are superior to mechanistic in regards to innovation.

According to Morgan, the organismic metaphor has contributed much to the theory and practice of organizational development and corporate strategy, which both are important areas.

The last strength Morgan discusses is that it focuses on ecology and interorganizational relations. To look at the interorganizational relations is important to be able to understand how the world of organizations evolves.

Weaknesses

One weakness of the metaphor is that it gives the impression that organizations and their environments are far more concrete than they actually are. In reality they are not that tangible but rather socially constructed phenomena. According to Morgan, the material aspects are the most important in organisms, whereas the immaterial aspects are more important in organizations. All the different variations of complex adaptive systems are not as concrete as organisms, but there still is a tendency to look at material aspects and lot of the inspiration comes from nature.

The metaphor might lead to a viewpoint of the organization being too much of a victim to the external world, where all it can or should do is to adapt. Morgan states that this view fails to recognize that organizations “are active agents operating with others in the construction of that world.” (p. 74)

Another weakness is the assumption of what Morgan calls functional unity. In an organism all elements normally work for the good of the whole, the system is unified and shares a common life and future. The parts of organizations do not normally function in such a harmony and unity as the parts of organisms do. Morgan states that “the emphasis upon unity rather than conflict as the normal state of organization may be an inherent weakness of the organismic metaphor.” (p. 75)

The last weakness mentioned by Morgan is the danger of the metaphor becoming an ideology. This has the implication that instead of merely being a metaphor, used as a tool for understanding, it also becomes guidelines used to guide how something should be. So if the elements of organisms are unified, the elements of organizations should also be unified. How big of a weakness this is depends on how you want to use the metaphor. In this thesis the metaphor has intentionally been used for more than just a way of understanding organizations.

Software Industry

There are some phenomena within the software industry that take more organic approaches than more traditional approaches, such as the Software Process Improvement movement. In this section we will look at two different approaches: agile methodologies, and Open Source communities.

Agile methodologies

Agile methodologies are a collection of methodologies for developing software. The term agile refers to that they aim at being able to swiftly respond to changes.

As mentioned before, this isn't something that could easily be achieved with a mechanistic approach. The methodologies were formerly labelled lightweight methodologies, since they all have in common that they try to minimize all excess luggage (which often implies little documentation). But since this is not a goal in itself, but a necessity to be able to respond fast to changes, the term was changed into agile, which more clearly states what they are aiming at. In the process of changing the term the proponents of the main agile methodologies (such as Extreme Programming, Scrum, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development, and Pragmatic Programming) put together the Agile Manifesto, which states:

> "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
>
> *Individuals and interactions* over processes and tools
>
> *Working software* over comprehensive documentation
>
> *Customer collaboration* over contract negotiation
>
> *Responding to change* over following a plan
>
> That is, while there is value in the items on the right, we value the items on the left more." (Beck et al, 2001)

Agile methodologies put a strong focus on the developers. They try to give the developers a good environment, support what the developers need, and then trust the developers to be able to deliver a good product. This is accomplished through tight interaction and highly motivated individuals who works in self-organizing teams. In a following chapter one agile methodology, Scrum, will be presented and the relation to complexity theory will be further investigated.

Open Source

Open Source communities emerge with a strong purpose from the beginning—to create a software product. These communities have strong cultures, in form of tradition of slang, history, and myths (Raymond, 2001). The membership is voluntary, it is even possible to start up a new project using the same source code that develops into something else than the original project. One thing that holds these, seemingly fragile, communities together is the culture and implicit rules. Raymond mentions two main sources of motivation for the members to contribute: doing the work because you need the outcome or love the challenge, and gaining respect from your peers. The leaders, or project owners, are quite humble—it is not regarded positive if the leader takes credit for the work of the community or even strives for being leader. Since there are no central authority telling the members what to do, they do what they are motivated to. But although the work seems unstructured Open Source communities have successfully created highly used software with high quality, such as Linux.

The similarities between Open Source development and complexity theory are quite strong. They both put a strong emphasis on the value of the culture, in order to retain the identity of the system. An Open Source project has one, very strong, purpose which everyone knows: to develop a specific piece of software. There is also no strong central authority commanding the developers what to do, but rather the work is coordinated through cooperation and what the developers want to do. Since everyone participating is doing so voluntarily there can hardly be anyone commanding the others what to do, instead people choose whether or not to follow the leader and even whether or not to continue working for the project. There is also a very open attitude towards information, since everyone outside the project usually have full access to what it is producing—especially the source code. From the outside an Open Source project might appear rather chaotic, and you might wonder if they can manage to produce anything, but somehow there is an order and they actually manage to sometimes produce software, sometimes with very high quality.

Summary

- The Butterfly Effect exists in aperiodic systems and implies that they are unpredictable. Nonlinear systems might seem disorderly, but it's an orderly disorder.
- Chaos is everywhere; the world is more nonlinear than linear. Most of the world is made up by nonlinear systems.
- Chaos theory is concerned with nonlinear systems, whereas complexity theory is concerned with complex adaptive systems. Complex adaptive systems are nonlinear systems but can sustain an identity through their actions and behavior.
- Complex adaptive systems consist of autonomous agents and all systems share the same characteristics and principles. In Holland's model these are aggregation, tagging, nonlinearity, flows, diversity, internal models, and building blocks. In Youngblood's model the characteristics are wholeness, connectedness, identity, balance, creativity, openness, and flexibility.
- The order in complex adaptive systems emerges at the edge of chaos. Here, rigor is not a goal, but rather a mechanism used to balance the system at the edge of chaos. If too little structure is used, the system will get too sensitive. If too much is used, the system will get too rigid.
- If a system gets too much information and loses its balance, it will go through a period of instability. If it manages to maintain its identity, it will self-organize into a new form that better copes with the new situation.
- Organizations can be seen as complex adaptive systems. The core ideology of an organization based on complexity theory is purpose and principles. They constitute the identity of the organization and helps aligning its parts.
- Since the agents in a complex adaptive system are autonomous, there is no central control. This implies that organizations, in the organic view, should rely more on leadership than control.
- Leadership is not for the few appointed leaders, but rather for everyone. Leadership is not downward-looking, but rather concerned with self, superiors, and peers. The need to lead subordinates is quite small, as long as they in turn lead self, superiors, and peers.
- The main strengths of the metaphor are that it emphasizes understanding of the relations between organizations and their environments, and that it aims for survival and innovation.
- The main weaknesses are that it tends to focus on material aspects, it might make the organization a victim to the environment, and it assumes functional unity.
- Agile methodologies aim at swiftly being able to respond to changes. They provide a good environment and are then trusting the developers, working in self-organizing teams, to produce good results.
- Open Source communities are formed of voluntary developers, sharing the purpose of creating software. The communities are hold together by culture, implicit rules, and personal motivation.

In the next chapter we are going to look at some different aspects of organizations in the light of complexity theory. By looking at these aspects we can realize some of the balancing acts that are important to consider in order to balance the organization at the edge of chaos.

CHAPTER 4

ORGANIZATION

In this chapter three different aspects of organizations are discussed in the light of complexity theory, namely the social structure, the physical structure, and knowledge management. Organizations are complex phenomena with many interesting aspects to discuss. The social and the physical structure are very related and they give many examples where you get the opportunity to consider how to balance structure and rigor on the one hand, and freedom and self-organization on the other hand. By looking at some cases it is probably easier to realize what implications complexity theory have on other cases, and to realize that there are few definitive answers.

The chapter begins with a discussion about the social structure of an organization from a complexity theory point of view. The areas that are of special interest are the organization as a whole, the supportive functions, and project-related issues, such as sub teams and roles. Next we look at how the physical structure, in form of organizational geography, layout, and symbolic implications, affects the social structures and collaboration. The chapter concludes with an overview of the five enabling conditions for organizational knowledge creation and how they relates to complexity theory.

Social structure

The social structure of an organization is very interesting to look at from a complexity theory point of view. It affects and is affected by self-organization in the highest degree. In this section we will look at the organization as a whole, the supportive functions, and the projects.

Organization

When you look at an organization, you must make some sort of division between the organization and the environment. This division is, however, not trivial. As stated by Hatch (1997): "[i]n fact, the organization itself cannot really be separated from its environment because it makes up the environment along with other organizations with which it is involved." (p. 73) Before the concept of organization was used—it was introduced with the Industrial Revolution (Gharajedaghi, 1999)—industrial work was organized through subcontracting (Hatch, 1997). To get a contract a group of people got together, often led by a master craftsman. With the industrialization the border of the organization became clearer since the organizations contained more of what was needed to get the job done—there was not the same need for subcontracting. In post-industrial organizations the organizational boundaries are disappearing. "This idea inspires views of a future in which organizations are much smaller, more fluid and flexible than they are now, with invisible or no boundaries between the organization and its external environment.” (p. 26) New organizational forms in the post-industrialism include network and virtual organizations as well as strategic alliances.

If there are no boundaries towards the external environment the organization is more likely to self-organize—people will probably migrate between adjacent organizations in order to reach structures suitable for the work at hand. With more clear and visible boundaries people will not naturally migrate between organizations, but they must rather be moved between the organizations. It is not necessarily a disadvantage to have some boundaries towards the environment—if the boundaries are too vague and fluid the organization might experience too much chaos and not be able to do any constructive work.

Supportive functions

The term project is often used to describe an effort with a limited and specified goal and scope, with limited resources, and with limited duration. Most of the work in software organizations is conducted in form of projects. There are, however, work tasks not directly related to any project, the supportive functions.

At IDX Systems, a healthcare software company, the whole organization is based on Scrum (Schwaber and Beedle, 2001). To coordinate related Scrum teams they use something that is called Scrum of Scrums, which basically means that key persons from the different teams form a Scrum used for coordination. At IDX the team leaders of each Scrum of a product line form a Scrum of Scrums and meet weekly. Each part of the organization is based on teams, including the management Scrum, which meet monthly.

At General Electric’s jet engine plant in Durham, GE/Durham, everyone besides the plant manager are technicians and are all part of a team (Fishman, 1999). Each team works on an engine, from the beginning to the end, and makes all the operational decisions. Everything in the plant that are not handled by the plant manager or directly by the teams are dealt with in form of councils. Each council consists of members from different teams, for instance the human resources council has one member from each team. The councils handle human resources issues, supplier problems, engineering challenges, computer systems, discipline, and rewards. To make decisions the councils always try to reach consensus, where the view of all the members has roughly the same weight.

One example of how to take care of supportive functions is an exceptional case where they needed to make some serious reductions of the expenses. To please her boss at General Electric, the plant manager quickly created a plan and sent it to him without letting anybody else know. At the same time she formed an expense council to educate the employees to be more cost-conscious. In three weeks the council had grasped the basics and developed a new plan that was much better than the plan the plant manager developed. The plant manager then sent the new plan to her boss. However, the expense council continued for six months with its “cost education”.

While projects come and go, the supportive functions need to be taken care of at all times. Furthermore, the nature of the work doesn’t usually change drastically but is rather roughly the same even if the rest of the organization goes through changes. This might indicate that there is not the same need for supportive functions to self-organize.

Project

The concept of projects is quite established and seems to suit the way software is developed since it is not routine and you must approach each case in a unique way. One issue about projects is how they ought to be initiated, or more specifically, if they should emerge and self-organize or if they should be created. This is the same kind of issue that was discussed above if organizations should emerge and self-organize. One main difference is that there are much more potential interactions when considering whether an organization should emerge since you have to consider the environment as well. This ought to make it more feasible for projects to emerge in an organization than for whole organizations to emerge from their environment.

At 3M they have something they call the "15 percent rule" (Takeuchi and Nonaka, 1995). It's a rule that says that everyone can spend 15 percent of their time to do something else than what they normally do—almost one day a week to pursue their dreams. It is during these 15 percent some of 3M's most successful inventions, such as the Scotch tape and Post-it Notes, have come to life. If you have an idea that you would like to realize you use your own 15 percent. You can also try to rally others to your idea and get them to devote their 15 percent to the effort (Poppendieck and Poppendieck, 2003). It is called bootlegging when you start a project without any approval or funding. As the work progresses and the idea is turning into something someone considers worthwhile, it might get "official" status. There are different degrees of "official": from getting the support from sponsoring managers who keep the project out of sight from the rest of the company since it's not strong enough, to "official" status with support from the company. To get the "official" status the product must fulfill three requirements: it must meet a real need, it must use 3M technology, and it must have a good profit potential.

Many organizations would probably have problems with letting the projects emerge and self-organize as they do at 3M. The reality for many projects is that they have a number of people at their disposal. This approach has a major impact on the number of potential interactions—from containing everyone in the organization, as in 3M, to just containing the ones assigned to the project. This implies that the possibility for self-organization is dramatically reduced as well. The approach at 3M is probably better suited for rather large organizations since the larger the company is the more possibilities there are that people with a common interest can get together and accomplish something substantial.

Sub teams

Through the aggregation mechanism a group of people (agents) self-organize into small groups (meta-agents). This implies that in a project the members normally form sub teams if there is a sufficient number of people. One problem you face when sub teams emerge is whether or not to acknowledge them and make them into official sub teams.

In Scrum they approach the problem by dividing the project into several teams, each consisting of five to nine persons (Schwaber and Beedle, 2001). There is one shared list of future work tasks for the whole project, but each team owns and maintains its own list of work tasks that it has to take care of during the current iteration. This implies that the teams can ideally work independent of each other. Key persons from the different teams meet on a ragular basis and coordinate the effort.

If the sub teams are turned into official teams the future possibilities for the project to self-organize into new constellations are rather reduced. It is, as always, a matter of applying the right amount of rigor in order for the system to balance at the edge of chaos.

Roles

Roles have impact on interactions in two, seemingly conflicting, ways. In one way they can open up for interactions, through the tagging mechanism—you know who to interact with based on their role. In another way roles might stifle interaction. If roles are used in such a way that they function as tools for division of labor, they are actively used to reduce the need for interactions (Schrage, 1995).

A role is either the characteristics and expected social behavior of a person, or a function or position. If used as a function or position, a role might lead to division of labor, reduced interactions, and thus interfere with the self-organization and collaboration. However, used as characteristics or expected behavior they might make collaboration and interaction run more smoothly, and lead to more self-organization.

Another issue with roles is how they are filled. If they are filled before any self-organization has taken place they restrict the self-organization. If they are used to maintain a self-organized structure, they hinder future self-organization.

In Scrum there are few roles outside the team (only Scrum Master and Product Owner, which will be described in the following chapter). Inside the team there are no official roles, not even to denote the functional specialization of the team members. Everyone is expected to do what they can to contribute to the goal. The presence of roles in the team is only in form of "characteristics" roles (such as skill, functional specialization, personality), which are used to guide the self-organization.

At the beginning of NBI (National BankAmericard Incorporated), what later became VISA, they used no titles (Hock, 1999). When newly recruited employees asked about titles they were given a long list of available titles they could choose among (such as Grand Duke, Lord, Lady, King, Queen, Ayatollah, Bishop, and Samurai). They could even come up with their own titles. However, there was one restriction—the title had to be used at all times. Just as no titles were used, the were no work descriptions. To answer the question of how people could know what a person did, Hock answered: "If what you do is not readily apparent to everyone by your words, conduct, and acts, that becomes a very interesting question." (p. 278) However, after a while they bent in to the pressure of being in "one of the most title-conscious industries that ever existed" (p. 278), and started using "manager" and "director" (which in time led to more titles). Hock believes that it was his fault since he used "president" to denote himself in the organization's bylaws.

At GE/Durham they have only one manager, the "plant manager" (Fishman, 1999). Although the plant manager is the only manager, a former plant manager states that she did only make ten to twelve decisions per year where she didn't consult the employees. Most decisions are made by the persons involved reaching consensus. On a daily basis the teams run themselves, mostly without input from the plant manager. The job of the plant manager is, on the operational level, to keep the employees focused on the goal of the organization. On the strategical level, the plant manager makes sure that the factory is working well as a whole. Moreover, the plant manager has to make sure that the plant has a good reputation, in order to get orders for new engines. The over 170 technicians have three different grades: tech-1, tech-2, and tech-3. These are in some way used as division of labor, since there are certain tasks that can only be done by tech-3, and some tasks that only tech-2 and tech-3 can perform. But this is a safety issue and anyone can reach tech-3 since the grade is based on what training you have completed. Despite there are some differences in what they are allowed to do, everyone cleans up.

Physical structure

The physical structure has a great impact on an organization. Two aspects are the organizational geography, and the physical layout, which both have implications on interaction. Another aspect is the symbolic implications of the physical structure.

Organizational geography If an organization is geographically distributed it will have serious restrictions on the interactions (which are essential for self-organization to take place). Although there exist plenty of means of communication, such as video conferences, telephone, e-mail, and various Internet solutions, they are not nearly as rich as direct communication. This might not be as devastating if the need for communication between the different sites are low, and the self-organization is mainly intended to take place within the different sites. One aspect that influences how the sites work, and how they interact is the local culture and lifestyle of the sites (Hatch, 1997). This might be positive, as it leads to more diversity, and negative, if it makes the communication more difficult.

Physical layout The physical impact of a workspace deals with how physical objects and the human activity are arranged (Hatch, 1997). The layout has a rather large influence on the communication and coordination of the workplace. This is especially true for large objects, like walls and large furniture, and for the people.

Cockburn (2002) mentions five different arrangements of two persons working together at the same site:

- Sitting side-by-side, using the same workstation (called pair programming in software development). There is nothing between them to obstruct communication and working on the same problem forces them to communicate.
- Sitting at separate, adjacent, workstations. There are very little that obstruct the communication, since it doesn’t take much energy to communicate.
- Sitting in the same room but rather far away from each other and facing different ways. It requires some effort for them to initiate and conducting the communication.
- Sitting in adjacent offices, separated by a wall. By being clearly separated, it takes substantial amount of energy to initiate communication.
- Sitting on different floors or in adjacent buildings. Since they are quite far apart it takes much time and energy to initiate communication, which certainly will lead to many lost opportunities of communication.

People “radiates” information all the time as they work, according to Cockburn. If they are close they can intercept more of that information than if they are far apart. This implies that there are clearly more information flows between the persons in the three first arrangements, where they are sitting in the same room. When a group of people is working in the same room on related tasks it is called an open workspace. Cockburn calls the unintentional kind of communication that takes place in a setting like this osmotic communication. Osmotic communication means that it happens unconsciously as people intercepts background sounds from others while doing something else. One concept that is important to take into consideration is what Cockburn calls drafts, which are unwanted information. Because of drafts, but also personal preferences, it might be important for people to have private places to which they can go to take care of personal matters and avoid osmotic communication and drafts.

A variation of an open workspace is a bullpen, where people working on non-related tasks, such as different projects, work in the same room. Since the tasks are very diverse the environment becomes rather drafty. Another variation is skunk works rooms, where related groups are placed close to each other with open doors between them. Skunk works rooms are reducing the drafts more than bullpens while preserving some osmotic communication, and it doesn't take too much energy to initiate communication between the groups.

Working close to each other makes the people form a community with a group identity and culture (Cockburn, 2002). This is very important to take into consideration in regard to self-organization. If people are working far apart, in separate offices, they might not self-organize that well. Working in an open workspace is probably better for the self-organization of the group. But there might be occasions when working close to together creates social structures not suitable for the work at hand. One example is a bullpen, where people working on different projects are sitting in the same room. In such a case there might emerge structures conflicting with which project they are working on.

Symbolic implications

Besides the direct impact of the physical structure there are also symbolic implications. According to Hatch (1997) it is important to "not overlook the importance of meanings and interpretations associated with organizational symbols and symbolic events, and many aspects of physical structure serve in this symbolic capacity." (p. 252) Hatch states that the architecture is the most obvious example of communicative power. Buildings become, over time, a representation of the organization, which guides the people in regard to what they feel and think of the organization. Since people attach certain meanings and interpretations to physical structure, they behave in a certain way at the presence of certain types of physical structure. For instance might an open office have the symbolic implications that the communication becomes more open.

Another important aspect of symbolic implications, especially in relation to complexity theory, is the influence on the identity. Hatch mentions two factors: spatial elements (locations, buildings, color schemes) and spatial relations (geography, layout, design and decor). One example she mentions to describe the great influence on identity is how important it is for an investment firm in New York City to have a Wall Street address. When you consider the identity it is interesting to look at three different levels: personal identity (status), group identity (territorial boundaries), and organizational identity (corporate image).

Organizations usually have a lot of status indicators visible in the physical structure. If you have a high status you might have better access to support facilities and equipment, your office might be larger with better furniture and so on. At the beginning at NBI they did not have personal offices—it was completely open with large windows so that anybody could see the view of the city and the nature (Hock, 1999). But since Hock got loads of visitors, one of the conference rooms became "his office". After a while the pressure from other senior people, requesting their own offices, led to that the space was cut into offices. Another aspect of symbolic implications Hock mentions, is that he insisted on one principle: no one should ever be cut out from full view of the outside. (This is, however, something that has not been preserved since he left, since there are now blind cubicles in VISA.)

To use physical structure to create group identity is something that seems to be in accordance with complexity theory. Certain physical elements can be used to create a shared identity through the mechanism of tagging, just as a banner can be used to rally the members of an army. Another aspect is that if people are together it might lead to a shared identity, especially if others are outside the territory. The fact that the communication and interaction are far more richer when a group of people is working closely together, as mentioned above, makes the identity even stronger. According to Hatch (1997) it is hard to say if boundaries give groups their strong shared identity, or if it is the strong identity that leads to boundaries. She states, however, that both forces might be at work.

The corporate image is the impressions of the organization, formed by others. Architecture, design, decor, logotypes and so forth are physical elements that influence the corporate image. The corporate image is important when the organization is interacting with other organizations and individuals (again using the tagging mechanism). The organizational identity is interrelated with the corporate image but is the experiences and beliefs of the members, and is strongly influenced by the physical structure. According to Hatch it is important that the group identity is not too strong since it can interfere with intergroup cooperation. If we look at complexity theory we see that the organizational identity plays a major part here as a balancing mechanism. According to Youngblood's (1997) model identity guides the agents (in this case the groups) balancing integration (considering whole) and self-assertion (considering self).

Knowledge management

Knowledge is something that has become increasingly important. According to Hatch (1997) the "post-industrial society is organized around the creation of knowledge and the uses of information." (p. 24) Since knowledge is that important, it is important for complexity theory to facilitate knowledge and knowledge creation. Takeuchi and Nonaka (1995) mention five enabling conditions for organizational knowledge creation to take place:

Intention

The intention is defined as "an organization's aspiration to its goals." (p.74) Intention is essential for organizational knowledge creation to take place at all, since it drives the whole process. By using the intention it is possible to judge the value of the created knowledge. The intention of an organization is usually conceptualized in form of strategies.

Intention is quite close to what Youngblood (1997) calls identity in his model. Both intention and identity means that there is something that holds the system, or the organization, together. This is also quite close to the tagging mechanism in Holland's (1995) model.

Autonomy

Everyone within an organization should be able to act as autonomously as possible. This has the benefits of increasing the chance of unexpected opportunities as well as increasing the individual motivation to create knowledge. "Original ideas emanate from autonomous individuals, diffuse within the team, and then becomes organizational ideas." (Takeuchi and Nonaka, 1995, p.76) According to Takeuchi and Nonaka, self-organizing and cross-functional team is a powerful tool for creating an environment where individuals can act autonomously, while still acting according to the intention.

One fundamental principle of complex adaptive systems is that they consist of autonomous agents. Therefore it is not possible to have a complex adaptive system without any autonomy. And just as intention plays an important part for the enabling conditions, identity plays an important part to balance the autonomy against what's good for the whole system.

Fluctuation and creative chaos

Fluctuation takes place when the external environment interacts with the organization. "When fluctuation is introduced into an organization, its members face a "breakdown" of routines, habits, or cognitive frameworks. . . . A breakdown refers to an interruption of our habitual, comfortable state of being." (pp. 78-79) Hence, such a breakdown might lead to questioning the validity of ones frame of reference. If this kind of reflection doesn't take place, fluctuation tend to lead to "destructive" chaos. But fluctuation is considered that beneficial that leaders sometimes intentionally introduce fluctuation into the organization to achieve a sense of crisis (this intentional chaos is called creative chaos).

Balance is one of the principles in Youngblood's (1997) model. Balance does not mean that the system is static. It means that the system is constantly fluctuating, within certain limits, resulting in self-organization. If the fluctuation exceeds the limits, the system lose its balance and go through a period of instability. The period of instability and the self-organization that takes place due to the fluctuation are both quite similar to the breakdown Takeuchi and Nonaka describe.

Redundancy

Takeuchi and Nonaka define redundancy as "the existence of information that goes beyond the immediate operational requirements of organizational members." (p. 80) This means that the members have more information than needed to perform their current work tasks. Redundancy of information has a number of benefits. Sharing redundant information is a way to share tacit knowledge—knowledge that is not easily visible and expressible. It is also a prerequisite for self-organization and the "principle of potential command"—that each part is equally important and has a potential of becoming leader. Another benefit is that redundancy of information helps people understand where their place are within the organization, since they have some knowledge of other parts. However, it is important to balance the redundancy, since too much redundant information can lead to information overload.

Two ways to obtain redundancy are "fuzzy" division of labor and "strategic rotation". Fuzzy division means that there are no clear-cut division of the responsibilities, but rather that there are some overlaps. Strategic rotation is when people have a chance to work in other, often quite different, parts of the organization.

Creativity in complex adaptive systems uses positive feedback to enable small disturbances to have a significant impact (Youngblood, 1997). This means that creativity is related to fluctuation. A prerequisite for creativity and self-transcendence is redundancy—if there are no redundancy, or "messiness", the system won't evolve and be able to perform new actions.

Requisite variety

The members of the organization must have a degree of diversity that matches the variety and complexity of the environment, according to Takeuchi and Nonaka. If the internal diversity is too low, the organization won't be able to effectively deal with its environment. The variety can be enhanced by "combining information differently, flexibly, and quickly, and by providing equal access to information throughout the organization." (p. 82) It is also important that everyone has a close access to the information. Two ways to enhance the variety are to have a flat and flexible organization, and to change the organizational structure frequently.

Diversity is part of Holland's (1995) model. This principle emphasize how important it is that there is a large diversity of agents, all filling different niches. It is also part of Youngblood's (1997) model, which states that there should be an openness towards information, diversity and interactions. If there is an openness towards these three parameters, there is also a potential for creativity.

Summary

- Before the Industrial Revolution the boundaries between organizations were fluid. During the Industrial Age the boundaries became clear. In post-industrial organizations the boundaries are again disappearing.
- The balancing act on the organizational level is between fluid boundaries, to facilitate self-organization of the organization, and clear boundaries of the organization.

- Supportive functions might not have the same need for self-organization as the rest of the organization. At IDX Systems the whole organization is organized as Scrum teams. At GE/Durham the technicians form councils to take care of the supportive functions.
- One balancing act at the project level is between self-organization of the projects and explicit creation of the projects.
- At 3M they use the 15 percent rule to start new projects which are unofficial until they are considered worthwhile and strong enough.
- Another balancing act at the project level is between self-organization into sub teams and explicit creation of sub teams.
- At the individual level a balancing act is between using official roles as a means for division of labor and letting unofficial roles, or characteristics, guide the self-organization.
- In Scrum there are just a few roles, and none of them is inside the Scrum team. At NBI they did not use any roles at all in the beginning. At GE/Durham there is one plant manager, and the rest of the technicians have three different grades based on their training.
- The organizational geography has serious implications on the self-organization of an organization. If the organization is geographically distributed, the interactions between the sites is radically reduced and not as rich.
- The physical layout, especially of large objects and people, influence the communication and coordination of the workplace. Working close together makes people form a community with group identity and culture.
- In a bullpen setting there are relatively more unwanted information compared to an open workspace. Working in separate offices eliminates almost all the osmotic communication.
- The physical structure has symbolic implications as well. Buildings become representations for organizations and certain structures induce certain kinds of behavior. The physical structure, such as different offices, is often used as status indicators.
- The physical structure is also strongly affecting the identity within the organization. Externally, the physical structure is affecting the corporate image and how people interacts with the organization.
- There are five enabling conditions for organizational knowledge creation: intention, autonomy, fluctuation and creative chaos, redundancy, and requisite variety. The enabling conditions are all related to complexity theory and supported by it.

Based on the description of complexity theory we are now going to look at Scrum and how they relate to each other. Scrum is an example of how complexity theory can be of pragmatic use. The intention is that, by exploring the example of Scrum, the concepts and ideas of complexity theory will become clearer.

CHAPTER 5

SCRUM

This chapter is going to present Scrum and its relation to complexity theory. Scrum is an agile and lightweight methodology, primarily used for software development. It uses iterative development with small iterations. As all agile methodologies Scrum aims at being able to quickly respond to change.

This chapter will start by providing some background of Scrum—where it comes from, the original concepts and how it evolved into Scrum. The background will be followed by a description of the lifecycle. There will also be a description of the different roles, and the team. This will be followed by a presentation of the characteristics of Scrum and its underlying values. Based on the presentation of Scrum and its characteristics, its relation to complexity theory will be investigated.

Background

The term “scrum” originates from rugby and is the strategy used for getting an out-of-play ball back into play (Schwaber and Beedle, 2001). The team in rugby is, just as in Scrum, adaptive, quick, self-organizing, and has few rests. Besides the terms “scrum” and “sprint” (which is the name of the short iteration in Scrum) Scrum is not widely influenced by rugby.

Scrum was first mentioned in relation to new product development by Takeuchi and Nonaka (1986). What Takeuchi and Nonaka are presenting in that article is a holistic approach to new product development. This approach has six characteristics:

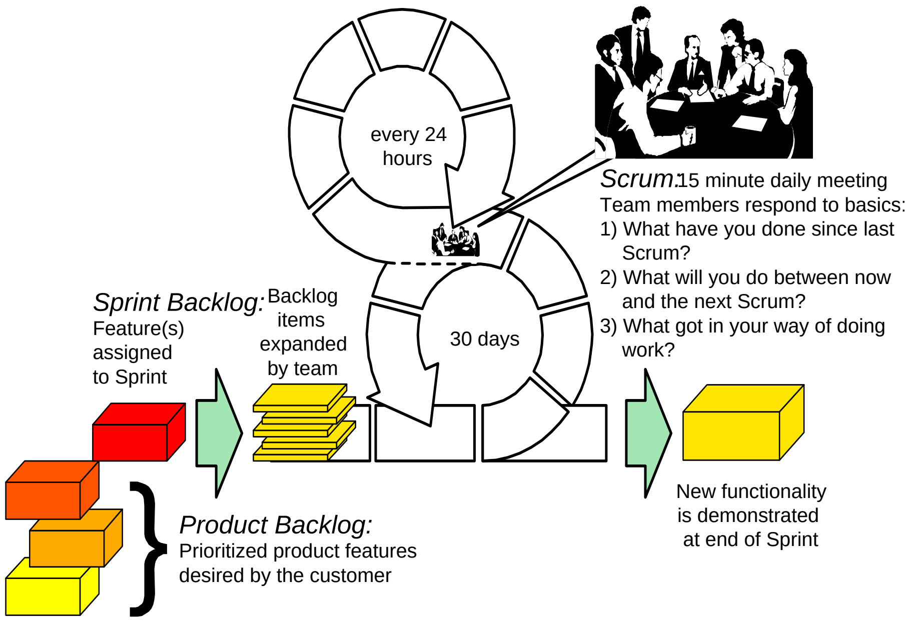
- Built-in instability—the team starts with a broad goal or general direction, which give the team much freedom. But at the same time top management sets demanding requirements, which creates a tension within the team.
- Self-organizing project teams—as the team comes to a state where old knowledge doesn’t apply, it experiences much fluctuation and ambiguity. At this state the team starts to self-organize—a dynamic order emerge and the team creates its own concepts. Takeuchi and Nonaka have identified three conditions for self-organization: autonomy (the team gets guidance, money, and moral support in the beginning but after that top management rarely intervenes), self-transcendence (the team is constantly striving for excellence), and cross-fertilization (members with diverse functional specialization, background and personalities all provide different perspectives).

- Overlapping development phases—instead of a sequential phases, which clearly separates analysis, design, implementation, testing and so forth, the team uses overlapping phases using iterative development.
- “Multilearning”—the team members are learning across two dimensions: multiple levels, such as individual, group, and organization; and multiple functions, learning things from other functional areas then their own.
- Subtle control—management try not to interfere with the team, since it’s autonomous, but management uses enough checkpoints to prevent the team from slipping into chaos. The emphasis is on “self-control”, “control through peer pressure”, and “control by love”.
- Organizational transfer of learning—the team members try to transfer what they have learned to others outside the team.

The article by Takeuchi and Nonaka was one of the main influences when Jeff Sutherland and Ken Schwaber, back in 1996, started formalizing what became Scrum (Schwaber and Beedle, 2001). They were also influenced from a number of other directions, such as complexity theory, software development practices at Borland and Microsoft, and process control theory. As Schwaber states “Scrum has become a major alternative to classic product development approaches. . . . Since its introduction, Scrum has been used in thousands of projects worldwide.” (p. 19) At IDX Systems, over 600 developers working on dozens of projects were using Scrum. It has been used in such diverse examples as software development projects, marketing programs, and business process reengineering (BPR) initiatives.

Lifecycle[1]

Figure 5.1 Scrum overview


every 24 hours
30 days
Scrum: 15 minute daily meeting
Team members respond to basics:
1) What have you done since last Scrum?
2) What will you do between now and the next Scrum?
3) What got in your way of doing work?
Sprint Backlog: Feature(s) assigned to Sprint
Backlog items expanded by team
Product Backlog: Prioritized product features desired by the customer
New functionality is demonstrated at end of Sprint

Product and Sprint Backlog

There are two artifacts that are used to hold the work tasks for a Scrum project, namely the Product Backlog and the Sprint Backlog. The Product Backlog contains the work tasks for the whole project, whereas the Sprint Backlog contains work tasks for just one Sprint. The Product Backlog evolves throughout the

1.Based on Schwaber and Beedle (2001)

project, whereas the Sprint Backlog is defined in the Sprint Planning. The items in the backlogs are prioritized—based on urgency—and can be just about anything, such as features, functions, technologies, enhancements, and bug fixes. Unless there are specific needs and requirements of how the different functionality should be realized, the items only consider what is needed and leave the realization specific aspects to the team to decide. Items with low priority might be rather vague, but as the priority get higher the more detailed description the items get.

Sprint Planning

The lifecycle of Scrum starts with two planning meetings, called Sprint planning meetings. In the first meeting the team meets with different stakeholders—such as customers, clients, and management—in order to elicit what to build in the following Sprint. The meeting starts with a presentation, by the Product Owner (refer to the "Organization" section below), of the highest prioritized items in the Product Backlog. The Product Owner then leads a discussion of which changes to the Product Backlog the participants consider necessary. The different aspects that are taken into consideration include what the team delivered at the end of the earlier Sprint, team capabilities, business conditions, and technology stability. The team decides, in cooperation with the other participants, which items they believe they could deliver in the following Sprint. When the participants have finished changing the Product Backlog they start working on coming up with a Sprint Goal. The goal is the objective of the Sprint and is to be used as a guidance for the team during the Sprint when making certain trade-offs. It is only a fairly short statement, about one sentence, thus no comprehensive description of what the Sprint will comprise.

When the Sprint Goal is set the second planning meeting takes place. This meeting is held by and for the members of the team. The objective of this meeting is to come up with the Sprint Backlog, which contains the tasks that are required to fulfil the Sprint Goal and the selected part of the Product Backlog. Other people than the team, such as Product Owner and certain specialists, might be invited to the meeting. However, it is crucial that they don't make decisions on behalf of the team, but rather are there to assist the team in their decision making.

Sprint

When the planning is done the Sprint takes off. The duration of a Sprint is usually 30 days. That the Sprint is only 30 days implies that no matter how bad the team performs during a Sprint, at most 30 days of work will be lost. Under this period the team works with the items in the Sprint Backlog and strives for achieving the Sprint Goal it has committed to.

Daily Scrum

During the Sprint the team conducts daily meetings, called Daily Scrums. These meetings are kept short and concentrated—they are supposed to take about 15 minutes. In order to be able to have such short meetings, they are restricted to three questions:

- What have you done since last Scrum?
- What will you do between now and the next Scrum?
- What got in your way of doing work?

These questions are to be answered by each member of the team. There are some room for quick decisions and responses, but anything that requires more than a really short period of time is deferred until later when only the affected persons need to be present. All team members are required to be present during the Daily Scrums. Anyone else is allowed to attend the meetings as well, on one condition: that they don't talk or interfere in some other way. If anyone would be allowed to speak during the meetings they would take up too much time and their usefulness would decrease.

Sprint Review

When the Sprint has come to an end it's time for a Sprint Review. The Sprint Review is where the team presents what they have accomplished during the Sprint to the Product Owner, management, and users. The team presents stories of what went wrong and what went right during the Sprint, as well as the strengths and weaknesses of the solution. A large focus is put on demonstrating the product. The performance of the team is compared to what it committed to for the Sprint—the Sprint Goal and the Sprint Backlog. The meeting takes about four hours and has an informal tone—there should not be more than two hours of preparation. Given the presentation the stakeholders are able to get a clear view of how far the team has come. With this new insight the stakeholders can much easier decide where to go next. When they see what has been accomplished they can easily come up with what they need next.

Organization[2]

Scrum Master

Despite not being mentioned above, the Scrum Master plays often an essential role in Scrum. The Scrum Master is probably the one who comes closest to a project manager—but the responsibilities and tasks are very different from what is normally considered being expected of a project manager. The Scrum Master lives and breathes to help the team. He or she is there for the team and not the other way around. “The Scrum Master is responsible for ensuring that Scrum values, practices, and rules are enacted and enforced.” (Schwaber and Beedle, 2001, p. 31) On a daily basis the Scrum Master get information from the team members of how they are progressing and what obstacles they are experiencing. It is then up to the Scrum Master to make sure that the obstacles are removed so that the team can continue with its work more easily. One reoccurring obstacle for a team is likely to be that the team have problems to agree on a decision. It is therefore important that the Scrum Master is able to make decisions for the team members when they ask for it. He or she also works with the management to ensure that the team get the support, the resources, and autonomy it needs. It might be hard for a team member to deny a request or order from top management, thus it’s important that the Scrum Master “protects” the team. He or she works like a firewall between the team and the chaotic environment. Besides removing obstacles the Scrum Master is also responsible to keep track of the progress of the team and inform others of the progress.

Scrum Team

The Scrum Teams are quite small, self-organizing, cross-functional, and autonomous. There are only about five to nine people in a team—to handle larger groups of people they are divided into multiple teams. If the team is too small, you lose some of the opportunity of synergy effects through rich interactions. If the team is too large the team will have a hard time to self-organize. And it is essential that the team can self-organize. There are no roles or imposed structure within the team—the team self-organize to best attack the work and the problems at hand. This way the team can be very flexible and adapt to a changing environment.

The fact that the teams are cross-functional is a necessity for the team to be able to reach the Sprint Goal. They must have most of the skills required during the Sprint, so that they can self-organize to the work without depending so much on outside help. This means that the team usually have to contain developers, testers, technical writers, and so forth. Once the team is assigned all members have to do whatever they are able to in order to reach the Sprint Goal they have committed to—there are no such thing as division of labor within the team.

As mentioned above, the teams are autonomous. The team should do whatever necessary to achieve the Sprint Goal, as long as it complies to the organizational standards and conventions. Moreover, during the Sprint nobody outside the team can tell it what to do. If the team realize after a while that they cannot fulfil all the items in the Sprint Backlog within the Sprint, they have to meet with the

2.Based on Schwaber and Beedle (2001)

Product Owner and the Scrum Master in order to identify what to remove without compromising the Sprint Goal. One aspect that shows the considerable authority the team has, is that it can terminate a Sprint if it realize that it won't be able to reach the Sprint Goal (management can also terminate a Sprint if, for instance, the Sprint Goal becomes obsolete). Although this authority is rarely used, it clearly shows everyone that the team is very autonomous.

Product Owner

The Product Owner is the one who maintains and controls the Product Backlog. He or she can be a project manager, a product manager or anyone else who is official responsible for the product. Everything is allowed into the Product Backlog—anyone can enter just about anything. But to keep the backlog manageable, the Product Owner continuously prioritize the items. Thus, things that probably never will be incorporated in the product get a really low priority. If anyone wants to change a priority he or she have to convince the Product Owner to do so. To succeed in prioritizing the Product Manager get help from the team and others with estimating how long the items might take to realize.

The Product Owner has, however, no control over the Sprint Backlog. It is the team that turns a subset of the Product Backlog into a Sprint Backlog during the Sprint Planning. It is also the team that maintains the Sprint Backlog during the Sprint—adding missing items and removing unnecessary ones. The main influence the Product Owner has over the Sprint Backlog is to assign high priority to those items he or she wants to be finished in the next Sprint, and low priority to those he or she wants to defer.

Characteristics

Empirical management

Scrum uses an empirical approach to manage time, cost, functionality, and quality (Schwaber and Beedle, 2001). The opposite to an empirical process model is a defined process model. In a defined process you can, based upon a given set of inputs, get the same output each time. This makes such a process repeatable and predictable, which allows you to conduct much planning and design up front. An empirical process model, however, concentrates on continuously gauge the performance and adapt the process in order to reach a desired result. The defined approach requires that the process is completely understood and repeatable, whereas the empirical approach acknowledge that the process is not fully understood and won't produce the same result given the same input. In Scrum the Daily Scrum provides feedback on a daily basis to the team, the Scrum Master, and others. This gives the team and Scrum Master an opportunity to adjust to how the work is progressing. On a higher level, the Sprint Review provides feedback on how the Sprint turned out to the stakeholders. Based on this information the stakeholders can make informed decisions of how to tackle the following Sprint.

There are some values that, according to Schwaber and Beedle (2001), emerge in an environment where Scrum is used. It is these values that are the foundation for all the practices. The values are: commitment, focus, openness, respect, and courage.

Commitment

In Scrum there is commitment on all levels. If no commitment exists, no work will be done since nobody will assign work to the team. The team commits to a Sprint Goal and a Sprint Backlog, and is then given the authority and autonomy needed to reach the goal. But the authority and autonomy are also accompanied with accountability—the team has to deliver and will be judged based on what it committed to. Commitment on the personal level is quite similar to the commitment of the team. The team members commit to what they will do the following day, and reports the outcome the following day on the Daily Scrum. Just as on the team level, the work is not assigned to a member and the members are accountable for the work they have committed to.

Focus

Scrum helps the team to focus on the problems at hand. When developing software there is often a never-ending flow of changes to the requirements. But all these requested changes are captured in the Product Backlog and might be implemented in a following Sprint. The team doesn't have to take these changes into regard during the current Sprint—it has its Sprint Goal and Sprint Backlog, and those are the only things that matters during the current Sprint. The Scrum Master plays also an integral part for the team to be focused. It is the Scrum Master that constantly reminds the team to be focused. Schwaber and Beedle (2001) quotes a Scrum Master who was really good at helping the team to be focused by always asking the same question at the Daily Scrums: "What's that got to do with code?" (p. 150). This shows the team that they are allowed to just focus at the current Sprint. The Scrum Master works as a firewall between the team and the chaotic environment. Moreover, the Scrum Master removes all the obstacles for the team, which also helps the team to focus.

Openness

The degree of openness and visibility is quite high in Scrum. The Product Backlog is visible for everyone, thus everyone can see where the project is heading. On a daily basis anyone can attend a Daily Scrum to get a good insight in how the team feels, how the work is progressing, and what problems they experience. The Scrum Master maintains a Backlog Graph, which shows how much work is left in the Sprint Backlog based on the estimations. This Backlog Graph shows clearly the trends of the Sprint to everyone interested. During the Sprint Review everyone can get a more fair and concrete picture of the process, since they can see the actual working software.

The high degree of visibility encourages the team to be honest. The team can't put up a nice front and hide the actual status. Everyone outside the team has access to the Backlog Graph, can attend the Daily Scrums, and the Sprint Reviews, which means that they can easily notice if what the team says doesn't harmonize with the picture presented at the Backlog Graph, the Daily Scrum and the Sprint Review. This also affects on the personal level, since the daily progress reporting at the Daily Scrums becomes much easier if the team members always report the actual progress.

Respect

Because of the self-organizing in Scrum it is important that the team members respect each other. They have different backgrounds and skills, especially due to the fact that the team is cross-functional. Since the team is self-organizing, the team has to respect and adjust to the different abilities of the members, in order to achieve the goal. The team commits to a Sprint Goal and a Sprint Backlog based on its abilities, thus the team is responsible to fulfill its commitments as a whole—if the team fails it is not the fault of an individual team member, but rather of the whole team. If the team respect and appreciate the differences between the members, the team can better organize itself to do its work.

Courage

Scrum is quite different from the traditional approaches that people are used to. Schwaber and Beedle (2001) mentions two types of courage the team needs: "the courage to find out that the environment will support [the Scrum] values, and the courage to be willing to find out that relying on one's own judgement is acceptable – even laudable." (p. 153) The courage they talk about is to have the courage to be determined to fulfil your commitments and to do the best you can. It is the courage to not give up when experiencing problems—to take charge of the circumstances and not let them rule you.

Relation to complexity theory

Scrum is explicitly influenced by complexity theory and is designed to benefit from what complexity theory has to say. It is also used as one of the main perspectives used when describing how Scrum can work (Schwaber and Beedle, 2001). There are some parts where the connection between Scrum and complexity theory is quite evident and others where the connection is somewhat more subtle. We are here going to look at the relation between Scrum and the models by Holland (1995) and Youngblood (1997), which were presented before.

Aggregation, Wholeness, Nonlinearity, Tagging

The members of a Scrum team can, by following the few, simple rules defined by Scrum, accomplish something greater than the sum of their individual abilities—the team obtains emergent properties which are impossible to predefine. The team members self-organize into different temporary, and perhaps some more permanent, constellations to accomplish the various problems they encounter. The dynamic structure of the team is, according to Schwaber and Beedle, typically pairs working together. There are thus agents (team members) working together creating meta-agents (pairs or small groups). And since the team is quite small and the members are working so closely together they make up a meta-agent. This meta-agent is collaborating with other agents (such as the Scrum Master, Product Owner) and other meta-agents (such as other Scrum teams).

For these relationships to emerge tagging is used. The people in the Scrum team uses the tags that imply team membership to self-organize into a meta-agent. The pairs and small teams use the tags that imply apprentice and master (to educate); domain experts (to make use of cross-functional expertise); or experts (to gain speed) (Schwaber and Beedle, 2001). When the team and others are collaborating, the tag for the other party might correspond to his or her role (such as Scrum Master or Product Owner).

Flows, Connectedness

In a Scrum environment there is a constant flow of information between the people involved—there is a large focus on communication. The team members are constantly communicating as part of the work. The Daily Scrums provide an excellent opportunity for communication to take place within the team and with the Scrum Master, as well as informing people outside the team. The team is communicating with the customer and the outside world on the Sprint Planning and the Sprint Review. The are also many other situations where communication takes place since it isn't something that is highly regulated—there are little focus on predefined, formal communication lines. Within the team they all have to communicate to be able to benefit by the other skills and to work as a group, which leads to a high degree of connectedness.

Identity

The Scrum team has a strong purpose, to develop the solution the customer needs. All other obligations and commitments are hopefully kept to a minimum. This helps the team to focus all its attention to the project at hand, and thus realize its purpose. The purpose is more granularly defined in the Product Backlog, as well as the Sprint Backlog. During the Sprint the team also have the Sprint Goal, which can help them creating a shared sense of purpose.

Diversity

In a Scrum team there are people with different focus and background, such as a Product Owner, a Scrum Master and developers. There is also a diversity among the developers, in terms of their skills and expertise. The diversity is even more stronger when the team is cross-functional (which it usually is) since it contains all the people necessary to complete the Sprint, such as programmers, testers, and technical writers.

Creativity

As mentioned before, the creativity in complexity theory is self-transcendence and uses positive feedback. There is a general focus on early and regular feedback. During the Daily Scrums the team, the Scrum Master and others get feedback on how the work is progressing and what obstacles there are. Another occasion, mentioned by Scrum, is the Sprint Review where everyone get feedback on what the team managed to accomplish during the Sprint. Hence, the question is if the feedback is positive or negative. I believe that the feedback is primarily positive. There are neither any detailed plans of when things are supposed to be worked on, nor any detailed process documents which describes how the work shall be performed—it is more important to do a good job than to follow plans and documents. There is, however, also some negative feedback. In the Daily Scrum the developers are supposed to have achieved roughly what they committed themselves to in the previous Daily Scrum, and the team are supposed to have fulfilled the Sprint Backlog it committed to.

Redundancy is a prerequisite for the self-transcendence to take place. In Scrum redundancy is primarily achieved through the fact that the division of labor within the team is "fuzzy"—the tasks are not clearly divided. Another issue that contribute to the redundancy is that the teams are cross-functional, and the members sometimes have to perform work tasks outside their functional specialization and have thus redundant information for the work task at hand.

Openness

It should come as no surprise that openness is something that is supported and emerge in Scrum as openness is one of the five values of Scrum. There is an openness towards all the three important parameters: information, diversity and interactions. Everyone is allowed and encouraged to share information with the others in the team and with people outside the team (during for instance the Daily Scrum). The information can be of any kind, such as the progress or how to solve a specific problem, which implies a high degree of diversity. There are also no restrictions for the interactions since they are allowed to emerge through self-organization.

Flexibility

Scrum is an agile methodology, which means that it tries to be as capable as possible to respond to changes in the environment. The Product Owner can at any time make changes to the Product Backlog and everyone else can make additions. The team is then able to respond to the changes in the next Sprint if they are considered important by the Product Owner. During the Sprint the team only has to focus on accomplishing the Sprint Goal and the Sprint Backlog. To be successful the team must be flexible in its organization and self-organize into structures suitable for the circumstances.

Internal models, Building blocks

I don't believe that internal models and building blocks are something that is specifically addressed in Scrum. They are mechanisms that the developers use as agents (individual level) and as meta-agents (team level). Schwaber and Beedle (2001) are however stating that the Sprint Goal, the Backlog, the requirements, and the architecture are internal models on the individual level, and shared models on the team level. But that doesn't seem to be the same kind of models Holland (1995) is talking about in his model. I think that ideas of how they should work in different situations, that is best practices, are closer to his description of internal models.

Balance

There are not anything in Scrum which is explicitly concerned with creating a fluctuating state. But there is a general acceptance towards fluctuation, both in terms of how the work is performed and the details of what is created.

Summary

- The approach to new product development by Takeuchi and Nonaka (1986) had six characteristics: built-in instability, self-organizing project teams, overlapping development phases, “multilearning”, subtle control, and organizational transfer of learning.

- Based on the approach by Takeuchi and Nonaka, Jeff Sutherland and Ken Schwaber started formalizing what became Scrum. They were also influenced by, for example, complexity theory, software development practices at Borland and Microsoft, and process control theory.
- The work tasks the team has committed to do during the Sprint is listed in a Sprint Backlog, which is maintained by the team. All other work tasks are listed in a prioritized Product Backlog, which is maintained by the Product Owner.
- During the first Sprint planning meeting some of the Product Backlog is presented, changes are made, the team selects the items they can commit to deliver in the following Sprint, and a Sprint Goal is established. In the second meeting the team creates the Sprint Backlog, which is supposed to fulfil the Sprint Goal and the selected Product Backlog items.
- During the Sprint, which is normally 30 days, the team holds short Daily Scrums where the developers tell what they have done, what they will do, and what obstacles they have encountered.
- At the end of the Sprint the team presents what they have accomplished in a Sprint Review. Their performance is compared to what they have committed to in terms of the Sprint Goal and Sprint Backlog.
- The Scrum Master lives and breathes to help the team, which often is to remove the obstacles that are reported or observed at the Daily Scrums. The Scrum Master works as a firewall, protecting the team from the chaotic environment.
- The teams are small, autonomous, and self-organizing. They are also cross-functional, which means that they have all the skills needed to reach the Sprint Goal within the team.
- Scrum uses empirical management, which means that it continuously measures the performance, and adapts the process in order to get a desired result.
- There are five values that emerge in Scrum: *commitment* on all levels instead of assignment; *focus* on the problems at hand in the current Sprint; *openness* and visibility into the team for everyone; *respect* of each other’s differences; and *courage* to do the things you have committed to and to do your best.
- Scrum is influenced by and explicitly designed to benefit from complexity theory. By following the few, simple rules of Scrum the team obtains emergent properties—the sum becomes larger than the parts. The team self-organize into different constellations, guided by the tagging mechanism.
- Scrum’s focus on communication and close collaboration leads to a constant flow of information. There is also a high degree of connectedness, each person involved is communicating with many others, since there are no predefined communication routes.
- The team has only one purpose, to develop a solution that the customer needs, which give the team a strong identity. The team is normally cross-functional, which gives the team a high degree of diversity—the team can handle everything that needs to be done within the Sprint.
- There are several different occasions where feedback takes place. The feedback is primarily positive and it is more important that the result is good than the plan is followed.

- Scrum is more oriented towards openness than restriction of the three aspects: information, diversity and interactions.
- As an agile methodology, Scrum is intended to be able to easily respond to change. Changes could be done at any time in the Product Backlog and are, if they are urgent enough, incorporated in the following Sprint.

CHAPTER 6

CONCLUSION

In this chapter a number of areas will be discussed: the limitations of complexity theory, why it is so hard to find good examples of "organic" organizations, the balancing acts, Scrum, the hypothesis and how it has been dealt with in this thesis, and what I think my contribution is with this thesis. Finally, some thoughts are provided on what further work is needed.

Discussion

Limitations

In order not to use complexity theory in the wrong way I think it is important to be aware of its limitations. As I see it, complexity theory is preferably used as an metaphor—that is, it can be used as a way to understand organizations and to guide the future development—and that has been the main focus in this thesis. As a metaphor it shares the same characteristics as metaphors in general and a few that is specific for this metaphor, which have been discussed earlier in this thesis.

If a critique against the machine metaphor is that it downplays the human aspects of organizations, the same can be said about the organic metaphor if used in the wrong way, for instance by overextending it. In the machine metaphor humans are compared to mechanical parts, which clearly is a bad match. But in the organic metaphor humans are compared to agents. Although agents are closer to humans than mechanical parts, I believe there is still quite many aspects of humans that are not covered. Can you really compare a human with an antibody or even an ant? Can you really expect a human to consistently follow a small set of rules?

Balancing

I have tried to communicate in this thesis that complexity theory is about a balancing act between too little and too much structure. There isn't one state that is optimal, but rather a fluctuating state where everything is between acceptable boundaries. And sometimes there needs to be periods of chaos in order for the system to survive and thrive long-term. For every decision when organizing I believe it is good to pose the question: will this lead to too much structure—thus interfering with the self-organization—or will it lead to too little structure—thus exposing the system to too much chaos. You need to be aware of the benefits and dangers of too much structure, as well as the benefits and dangers of too little structure. I think that the advise Highsmith (1999) put forward is good: "[a] little bit less than just enough".

Scrum

Scrum is what initially led me to complexity theory and it has been used as a example of how to apply complexity theory to software development. To me it seems as a good tool and a good set of guidelines to use when performing the balancing act. Since the mechanistic worldview is so dominating, good tools are needed to provide the right support. There are probably many different tools available that are worth looking more into. One place to start looking is probably other agile methodologies.

In Scrum they have chosen to not let any chaos into the team during the Sprint, but between the Sprints and outside the team chaos is allowed. That seems to be a good balance between too much and too little structure. The developers can concentrate on one piece at the time which doesn't change, and the customers have the possibility to change their minds at all times as they learn more about their own needs and what is possible. There are also a number of other rules or guidelines that affects the balance, that seems to be well adjusted for most software development. But as all tools Scrum has it limitations and it is important to learn in which situation it is well suited.

Where are the "organic" organizations?

It is rather easy when you talk about (conflicting) worldviews that you see everything in black or white. In theory the mechanistic and the organic worldwiew are opposites. But in practice, in the real life, there are probably more different shades of gray. People, and organizations, have worldviews influenced by both (and probably other) views, and might apply them to different extent in different contexts—for instance, a manager might see things differently when he or she is at home than at work.

It became really obvious to me that the world is more gray, than black and white, when I was trying to find companies based on each views for this thesis. Companies that appeared to be based on a mechanistic worldview, wasn't so "bad" in reality, and vice versa. However, it shouldn't be that surprising if you are agreeing with what complexity theory has to say. The reality is indeed very complex. And since organizations consist of many different people, each with its own background, opinions and ideas, the organizations are bound to have a blend of worldviews.

The difficulties I had finding an "organic" organization really made me rethink if complexity theory is useful in real life and not just an idealistic theory. There are loads of literature about the subject, so why are the organizations so hard to find?

There are, of course, the standard reasons why organizations have problems adopting new ideas and concepts, especially when they differ so much from what is conventional. And I think that the current strained economy makes people tend to mainly stick to conventional methods. You usually don't have slack reins when the times are tough, instead you have to be able to show that the changes to the organization you are proposing have the desired effect. To me it seems that applying complexity theory on an organization at a large scale is still quite risky. I think that more examples of successful applications of complexity theory is needed, and some way of quantitatively showing what kind of result you can expect. But I suspect that it can be quite challenging to do since complexity theory in itself is very general, it doesn't prescribe any details of how to perform the work. What you can do, more easily, is to take specific models that are influenced by complexity theory in a high degree, such as Scrum.

I suspect that another consequence of the strained economy is that slack is, more than usually, considered being waste, which is an idea that fits better into the mechanistic than the organic view. Such a thing as 3M's "15 percent rule", for example, is probably quite hard to introduce in a company that have financial problems. But it's not just a problem with this example—if there is a general skepticism towards slack and you try to streamline the organization and reduce any variations, you apply negative feedback and reduce the potential for creativity. But what you want is positive feedback, that is to amplify small disturbances, according to the model by Youngblood (1997), which was discussed earlier.

Another factor that I believe plays a significant part, that doesn't necessarily depend on the current state of the economy, is there seems to be skepticism towards some of the concepts put forward by complexity theory. This isn't something that I have scientifically researched, but something I have noticed when talking to others, both inside and outside the software industry, about this thesis or some elements of it. The main issues seem to be if it really works to lose the traditional way of controlling, and the notion that everyone should be a leader. I think the main reason for this is that complexity theory is part of a different worldview than the worldview of most people. It is obviously a great challenge to introduce these things if it requires people to change their worldview in order to be able to accept the new concepts.

Hypothesis

The hypothesis for this thesis is:

> Complexity theory contains elements and aspects that can be applied on software organizations—it is a meaningful metaphor. By studying complex adaptive systems it is possible to derive new ways of organizing software organizations in a way that better adapts to the ever-changing environment than current, mechanistic, organizations. By emphasizing self-organization, autonomy, leadership and collaboration it is possible to create organizations that are well suited to live in today's complex world.

It is quite hard to say if the hypothesis still holds, but that depends much on the nature of the hypothesis. You can never prove a metaphor on its own, you can only determine if you can make use of it in a particular situation and anticipate for which other situations it is applicable. For me, personally, the metaphor has made sense and have been useful when I have worked with this thesis, as well as in other, completely unrelated, situations.

But, as I have shown in this thesis, it is also a pretty clear that Scrum is highly influenced and compatible with complexity theory. Almost all elements of the models of Holland (1995) and Youngblood (1997) are more or less directly addressed in Scrum. Since they match each other so well and since Scrum has been proven successful in many software developing efforts, it shows that complexity theory is of some use in software development, and not just a theory that sounds good.

IDX Systems, GE/Durham, 3M, and VISA are all real life examples of organizations used throughout this thesis where they have used new ways of organizing complexity theory. These should not be seen as prescriptive, in the sense that their ways of organizing must be applied on organizations trying to go for a more organic approach. Instead, they should be seen purely as examples or sources for inspiration of how you could do. The idea is that new ways of organizing should appear that suit the current situations through self-organization and through the balancing act. To know if the new ways of organization is better than the traditional for each situation is of course hard to know in beforehand, and this thesis can neither prove nor disprove that. That is also the case with the last sentence in the hypothesis. But even if it cannot, based on this thesis, be proven or disproven I still believe that it is true.

Contribution

It has been my intention with this thesis to provide you, as a reader, with a good insight into complexity theory. It can be a rather fuzzy and complex subject with few clear answers, that's why I have tried to paint a clear picture of complexity theory from at least one perspective.

Furthermore, I have tried to highlight certain areas where complexity theory affects organizations and some balancing acts that is important to consider. One large part of this has been Scrum, which presents a number of areas and balancing acts.

Finally, I have provided a number of specific examples of the consequences complexity theory might have on organizations. Even here Scrum has played a large part, and has been presented as one possible concrete application of complexity theory. I have both shown that Scrum has strong relations to complexity theory, and that complexity theory really can be used in real life.

I hope that you now have enough information to know if you want to explore complexity theory further, or if it isn't a tool that suits you.

Further work

This thesis has been very theoretical in nature. Much more work is needed to study the application of complexity theory in practice. Since the mechanistic worldview is so influential it might probably be some resistance to introducing something more organic in an organization. That's why I think it could be really important to study people's attitudes towards complexity theory and what could be done to reduce any possible sources of resistance to change. One way is probably to study what effect it has and how much can be gained, if anything. And finally, I believe that more work is needed to find various concrete ways to realize the concepts and theories.

REFERENCES

Beck, Kent et al. 2001. *Agile Manifesto*. Online at: http://www.agilemanifesto.org

Bennatan, E.M. 1994. *Software Project Management: A Practitioner's Approach*. 2nd ed. McGraw-Hill.

Cockburn, Alistair. 2002. *Agile Software Development*. Addison-Wesley.

Fishman, Charles. 1999. *Engines of Democracy*. Fast Company. pp. 174-. Issue 28. Gruner + Jahr USA Publishing.

Gharajedaghi, Jamshid. 1999. *Systems Thinking: Managing Chaos and Complexity*. Butterworth-HeineMann.

Gilb, Tom. 1988. *Principles of Software Engineering Management*. Addison Wesley.

Gleick, James. 1988. *Chaos: Making a New Science*. Heinemann.

Hatch, Mary Jo. 1997. *Organization Theory: Modern, Symbolic and Postmodern Perspectives*. Oxford University Press.

Highsmith, Jim. 1999. *Beyond Optimizing*. Software Development Magazine. September 1999.

————, James A. 2000. *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. Dorset House.

Hock, Dee. 1999. *Birth of the Chaordic Age*. Berrett-Koehler Publishers.

Holland, John H. 1995. *Hidden Order: How Adaptation Builds Complexity*. Addison Wesley.

Humphrey, Watts S. 1997. *Managing Technical People: Innovation, Teamwork, and the Software Process*. Addison Wesley.

Morgan, Gareth. 1986. *Images of Organization*. Sage Publications.

Paulk, Mark C. et al. 1995. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison Wesley.

Poppendieck, Mary and Poppendieck, Tom. 2003. *Lean Software Development: An Agile Toolkit for Software Development Managers*. Addison Wesley.

Raymond, Eric S. 2001. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly & Associates.

Sanders, Irene T. 1998. *Strategic Thinking and the New Science: Planning in the Midst of Chaos, Complexity, and Change*. The Free Press.

Schrage, Michael. 1995. *No More Teams!: Mastering the Dynamics of Creative Collaboration*. Currency/Doubleday.

Schwaber, Ken and Beedle, Mike. 2001. *Agile Software Development with Scrum*. Prentice Hall.

Standish Group. 1998. *CHAOS*.

Takeuchi, Hirotaka and Nonaka, Ikujiro. 1986. *The new new product development game*. Harvard Business Review. pp. 137-146. January 1986.

————————————————. 1995. *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press.

Weinberg, Gerald M. 1986. *Becoming a Technical Leader: An Organic Problem-Solving Approach*. Dorset House.

Wheatley, Margaret J. 1999. *Leadership and the New Science: Discovering Order in a Chaotic World*. Berrett-Koehler Publishers.

Youngblood, Mark D. 1997. *Life at the Edge of Chaos: Creating the Quantum Organization*. Perceval Publishing.

Zahran, Sami. 1998. *Software Process Improvement: Practical Guidelines for Business Success*. Addison-Wesley.