Master Thesis
Software Engineering
Thesis no: MSE-2002-20
June 2002

Improving process efficiency

- A case study

Author: Nina Dzamashvili

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

Author Contact Information:
Nina Dzamashvili
Övre Brunnsvägen 46,
SE-372 36 RONNEBY
Sweden
E-mail: nina.dzamashvili@bth.se

University advisor(s):
Conny Johansson
Department of Software Engineering and Computer Science
Blekinge Institute of Technology
Box 520
SE-372 25 RONNEBY
Sweden

Abstract

Almost everybody in the software engineering world has read about or has heard of the importance and benefits of the mature software process. This master thesis aims to answer the question why despite all the efforts; software organizations still have difficulties to achieve the mature and effective software process. The results of the literature survey and a case study performed at a real software company have shown that there is a set of factors that may cause process inefficiency in the software companies. These factors are connected with the soft aspects of the software process such as motivation of the software engineers, process understanding and the process training.

Keywords: Software process, Software process improvement, Mature software process, Effective software process.

Table of Contents

Chapter No 1. Introduction

This chapter will present motivation for the research, as well as research questions and research goals. The reader can also find information about the structure of the document and some reading guidelines.

1.1. Background and motivation

Considering the number of the books that has been written on the subject “Software processes”, the amount of research and experiments that have been done in the given area, one may question if there is a need for hence another work, the master thesis, that is concentrated on the research of software processes and how to become the process oriented, mature organization.

The reason that made it attractive to write the master thesis on the subject of software processes, is the course named “Software Project and Quality management”, which was held in the autumn 2000, at Blekinge Institute of Technology, Sweden.

One part of the course was an experiment where a group of students, including myself, were given an assignment to interview software companies to find out their general software project and quality management practices. Each of the students was to interview three companies.

For the purpose of making the interpretation of the interview results easier the same questionnaire was used for each interview.

After the analysis of the experiment was made, we found a quite unexpected result. Seven out of 15 interviewed companies had no defined software process, two companies were working on defining their process, and six companies claimed that they had a defined organization wide software process. The result of the experiment, together with company names that were interviewed can be found in the table below.

Company Name	**Size (empl.)**	**country**	**organization’s defined Soft. process**
Martinsson IS	250	Sweden	Under development
Microdata	50	Germany	No
Datasave	120	Germany	No, but ISO certified
B-soft	8	Germany	No
Computer Technik Buchwolz	11	Germany	No
Komtel	240	Germany	No
NS-group	60	Sweden	Not documented
Maersk Data	1577	Denmark	No
Flextronics	1500	Sweden	Yes
Audiodev	100	Sweden	Yes
Tekko	22	Germany	Yes
Symbian	70	Sweden	Yes

Ericsson Software Technology(BGw)	40	Sweden	Under development
Micronet AB	20	Sweden	Yes
Cap Germini Ernst & Young	1500	Sweden	Yes

The demonstrated outcome of the experiment has generated a question that might be interesting to find an answer to:

Why do 7 out of the 15 interviewed software companies still have immature processes?

It is interesting to find out why, despite large amount of scientific works, which show the importance and the effect of mature software process on overall performance figures, software organizations still have immature processes.

Having the presented problem in mind we have started our investigation. During the first phase of the study no hypothesis were formulated, instead a decision was made to run an exploratory case study at some software company with the goal to study their software process, try to find the problems with their process and come up with a process improvement idea. The finite goal was to identify interesting problem areas for the further research.

Luckily, the management of department "ABC" at one of the large telecom companies in Scandinavia showed interest in the work. (The detailed description of the study is presented later on, in chapter 5).

From the beginning the company managers supposed that they were missing some information about their processes, so they expected to receive a detailed enough description of the practiced software process at the department and also the list of problems in their software process and an improvement proposal.

While working on the case study, as the knowledge about the software process in the company was gathered, a couple of factors that could negatively affect the quality of the process for the organization were identified.

Since we had a suspicion, that knowing about negative factors would help "ABC" to improve their process, the decision was made to focus the master thesis on answering the questions that are presented in the following section.

1.2. Research questions

The research questions of this study are constructed as follows:

?? What are the characteristics of the mature software process?

?? What are the features of the effective software process?

?? What is the connection between the mature and the effective software process?

?? What are the factors that negatively affect effectiveness of the software process in software organizations?

?? How can software organizations avoid those factors?

We believe that by answering those questions we can help software companies, both the ones that are trying to define their software process, and the companies that already have a defined software process, but are trying to reach high levels of the process effectiveness.

1.3. Research goals

We have identified a set of the goals that we have to accomplish in order to find answers for the research questions. Those goals are:

1) To review how mature and effective processes are characterized in the theory;
2) To find the connection between the mature and the effective software process;
3) To find out what factors are identified in the theory as positive and negative factors for the effective process;
4) To find out what are the negative factors for the real software company;
5) To come up with suggestions describing how to avoid the negative factors;
6) To compare the negative factors described in the literature with the negative factors found in the case study to see if there are any similarities.

1.4. Research approach

This investigation will be using the qualitative research approach, due to the nature of the data that research findings will be based on.

It is expected that the given investigation will be using qualitative data, such as software engineers' opinions and feelings about the software process that they are participating in. This information is needed to understand how the software process works at the studied software company.

Qualitative research approach was chosen as the main research strategy due to following reasons:

Reason 1: As discussed in the section 1.4.1 qualitative approach is suitable to analyze qualitative data.

Reason 2: We will not be able to collect the amount of data necessary to conduct realistic quantitative analysis.

1.4.1. Qualitative and quantitative approach

This section presents an explanation about qualitative and quantitative methods and also discusses when to use qualitative and when to use quantitative method.

Difference between qualitative and quantitative methods

According to [Strauss et al. 1998], qualitative research is the type of research that produces findings that are not necessarily based on statistical procedures, or other means of quantification, even though a scientist that is carrying out the qualitative research can code research data in a way that allows it to be analyzed by statistical methods.

One noticeable difference between those methods is how data is analyzed in order to form a theory. While speaking of qualitative analysis, one is aiming not to quantify the qualitative data, but rather to use a nonmathematical process of interpretation, carried out for the purpose of discovering concepts and relationships between the collected data, and then organize it on a theoretical explanatory scheme.

[Hoepfl97], in her article which is published in Journal of technology education, describes the difference between qualitative and quantitative inquires as follows:" where quantitative researchers seek causal determination, prediction and generalization of findings, qualitative researchers seek instead illumination, understanding and extrapolation to similar situations."

[Patton90] is presenting the difference between the mentioned research methods depending on how each of them approaches the studied subject.

According to [Patton90], the quantitative research uses probability sampling, which depends on the selection of a representative sample from the larger population. This is done in order to be able to generalize the research findings to the large population. By contrast the qualitative research uses different technique, so called purposeful sampling where sampling seeks information reach cases, which can be studied in depth.

[Svensson et al.98], also try to define differences between the two research methodologies. According to them, the quantitative method seeks answers by searching for similarities through statistical evidence within the set of the objects, which create the subset of the complete set of the same object type. When such a research finds relationships between the objects, one concludes that this is valid for all objects of the same type as the ones that were investigated. The qualitative research on the other hand seeks answers by researching different features of the single object.

The important aspect that [Svensson et al.98] stress is that the qualitative approach tries to find unique qualities of the studied object. This means that in qualitative approach each object occurrence is considered to be qualitatively unique.

Quantitative research on the other hand is forced to ignore diversity of qualities in the studied objects. The quantitative method, to be able to reason quantitatively, must assume that every object occurrence has the same qualities.

To make their point clear [Svensson et al.98] present an example, where two groups of scientists are studying the differences between two types of birds, namely a Swan and a Sparrow.

In the case of quantitative research, the two kinds of birds would be viewed as having the same set of qualities (properties). The difference could be identified only in numerical measure of each quality. For example, after studying certain amount of the Swans and the Sparrows, scientists can claim that the Sparrows are more intelligent than the Swans.

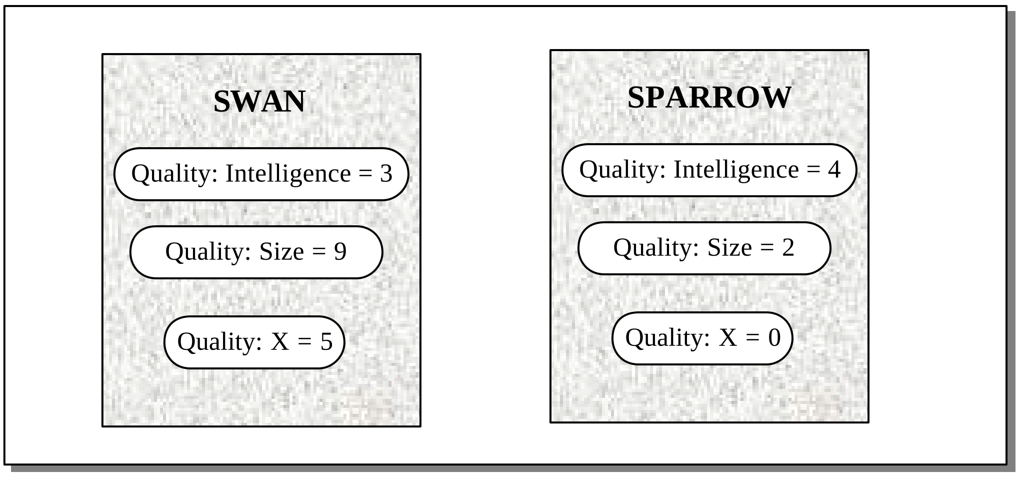


Figure 1: This picture shows the difference between Swans and Sparrows identified by quantitative research method.

In the qualitative method though, different qualities are not numerated and the quality sets are not the same. The Swan and the Sparrow are considered to have common qualities and also some qualities that are unique for each of the studied type.

This characteristic of the qualitative research enables scientists to explore new phenomena and find new information about already known objects. The figure below presents how the qualitative approach will view the difference between the Swans and the Sparrows.

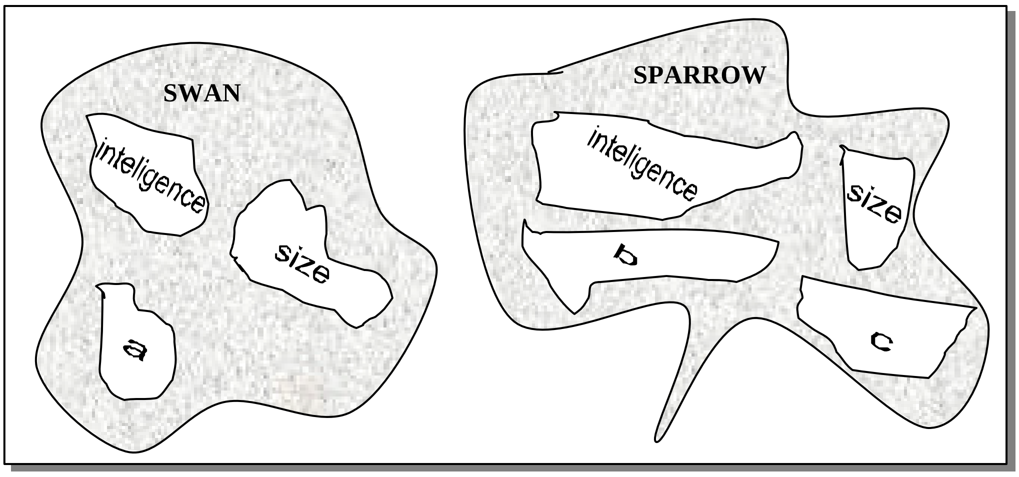


Figure 2: This picture shows the difference between Swans and Sparrows identified by qualitative research method.

The discussion about differences between qualitative and quantitative methods is continued by [Robson93]. In the book "Real world research", [Robson93] describes qualitative and quantitative approaches as hypothesis generating and hypothesis testing techniques.

A qualitative study is often regarded as the hypothesis generating study, because of the fact that while performing one, theories and concepts tend to arise from the conducted inquiry. On the contrary, the quantitative approach is started with generating and testing hypothesis. If a suggested theory is a good one, then it is supposed to cover a large number of the events and predict events that have not yet occurred or been observed.

[Hoepfl97], who writes that the qualitative research is the kind of research that produces but do not test the hypothesis and the theories, also supports [Robson93]'s approach. The qualitative research is an interpretative type, which means that conclusions made in that study can be based on researchers' experiences and interpretations. [Hoepfl97] thinks that later on, hypothesis suggested by the qualitative study can be backed up with quantitative research.

How do we know which method to use?

So how is one supposed to decide which research strategy is the best to use?

[Robson93] suggests that when we are designing the study, we should consider what kind of data we would have to deal with in our investigation. Data can be of two kinds, namely qualitative and quantitative.

Quantitative data:

Quantitative data is the data that is represented by the numbers [Robson93]. Using quantitative data analysis methods you could draw conclusions from that data, by simply logically organizing them into the tables or applying complex statistical analysis.

Qualitative data:

In opposite to the quantitative, qualitative data is represented by combinations of words [Robson93]. Qualitative data is called so because words have the capability to include richer context in comparison to just numbers, which are represented by quantitative data. Qualitative data is difficult to analyze with quantitative methods, since the qualitative data is not pure numbers. Nevertheless, qualitative data can be analyzed using methods referred to as qualitative analysis. There exist different qualitative analysis approaches.

Some of them are discussed by [Strauss et al. 1998]. References to the qualitative analysis methods can be found in [Robson93] also.

[Strauss et al. 1998] advocate that the qualitative research is applicable in situations where the research question is concerned with such soft aspects as human behavior, experience, feelings and so on, in other words where it will be difficult to analyze the collected data using statistical methods.

[Strauss et al. 1998] also claim that qualitative methods can be used to better understand any phenomenon about which little is known yet.

Qualitative method can also be used to gain the new perspectives on things about which much is already known, or to gain more in-depth information that may be difficult to convey quantitatively. [Hoepfl97] supports [Strauss et al. 1998]'s opinion that qualitative methods are appropriate in situations where one needs to identify the variables that might be tested quantitatively later on, or where the researcher has determined that quantitative measures are not adequate.

After presented discussion on qualitative and quantitative methods and their usage, one should still mention that many scientists advocate using a combination of discussed methods where possible.

1.4.2. Research road-map

In order to ensure the success of the thesis; the research will follow the road map that is shown below:

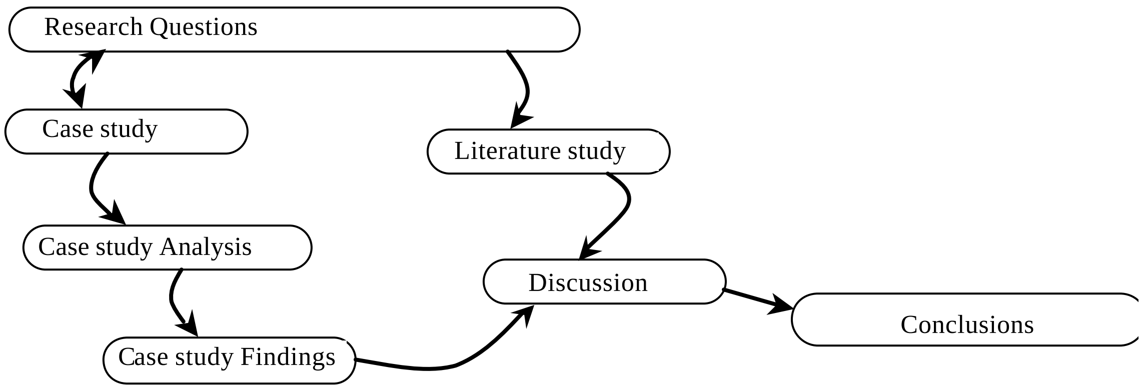


Figure 3: This figure presents how the research will be carried out in the master thesis

As one can see from the figure above, we are going to use a combination of two research strategies, a literature survey and a case study.

Literature survey

By means of the literature study we are going to gather information and knowledge about mature and immature software processes, effective software processes, and connection between the mature and the effective software processes. Thus, by doing literature survey we are trying to reach first tree goals of our research.

Exploratory case study

The exploratory case study will be used in order to accomplish goal number four and five, which is to find out what are the factors that negatively affect software process maturity/effectiveness in a real software company and to suggest what a company should do in order to avoid identified negative factors.

Discussion

After we are finished with the literature survey and the case study analysis, we will have a discussion session where we will compare the results from the literature and case studies. This part of our work is to address goal number 6.

Conclusions

In this part we hope we will be able to summarize the results of the study, in order to provide our conclusions and suggestions about the initial research questions.

1.5. Research scope and limitations

One can imagine the number of factors that can have an effect on software process efficiency. Those factors can be identified in connection with how the software process is engineered, how effective are the methods used by the software process, how the defined software process is executed and so on.

This master thesis though is primarily focused on execution of an already defined software process, and tries to investigate how software companies can make sure that the defined software process is followed and practiced in the software projects.

It should be noted that due to the limited time budget of the master project (400 hours), the discussion on how to quantify the effects of the discovered negative factors on the software process performance is out of the scope of this thesis and can be considered as a suggestion for the future work.

Chapter No 2. About Processes

This chapter is dedicated to describe the term "software process" and to present and explain why "software process" is one of the important concepts in science of software engineering. We will also present different levels of software processes in the software organization and the connection between them.

2.1. What is a process?

When one is talking about the process or processes, there is a clear need that one defines what he or she is meaning by it. This is because the term "process" can be interpreted or even defined in various different ways. For example, some people while talking about the software process really mean software development method, and others mean software development environment. [Ost87] on 9-th international conference on software engineering have even suggested that software processes are software too.

In the literature we often meet word combinations such as *Business process*, *Organization's common process*, *Software process*, *Development process* and so on. Do those phrases have completely different meaning, or are they somehow related to each other? It seems like we are not the only ones that is asking this question.

In their paper "Process diversity in software development", [Lindvall et al. 2000] are trying to answer the same question. "Is the process the way the company operates – from marketing to human recourses - or the way a developer produces design or code, or tests the software?"

Some people argue that depending on the perspective we can talk about the processes for the entire organizations, teams, or the individuals.

Since there obviously exists a large spectrum of the different types of processes, we decided to examine process diversity in more detail. We believe this will increase our understanding of the processes, and also help us to find the place of "Software process" among the ocean of other different kinds of processes.

2.2. Process diversity in an organization

Webster's dictionary defines the term process as "a system of operations in producing something… a series of actions, changes or functions that achieve an end result"

We can consider this definition as a common property for any kind of process. By this we mean that any kind of process can be considered as a series of identified actions and functions that lead us to the expected result. For example, making a cup of tea is a process the expected result of which should be a drink called "Tea".

What is making the processes so different from each other then? Why were we discussing process diversity in the previous section? One can describe a difference between two processes in terms of process result, and process perspective.

For example, in his book "Process mapping techniques", [Hunt96] presents different process categories. A given process can be associated to one or another category depending on the result of the process.

A process that results in a product or service that is received by the external customer is categorized as *Customer process*. A Process that does not result in a visible product, but is essential for an effective business management is referred to as *Administrative processes*. Another category of process, *Management processes*, includes actions managers should take to support the business process. Management processes include activities like goal-setting, planning, performance feedback, rewards and resource allocation.

Depending on the process perspective (Process organizational level), processes can be associated with one of the process levels that build up a process hierarchy tree. The lowest level of this hierarchy is an individual process; the highest level of the hierarchy is represented by an organization's business process. The picture below represents our view on the process hierarchy tree:

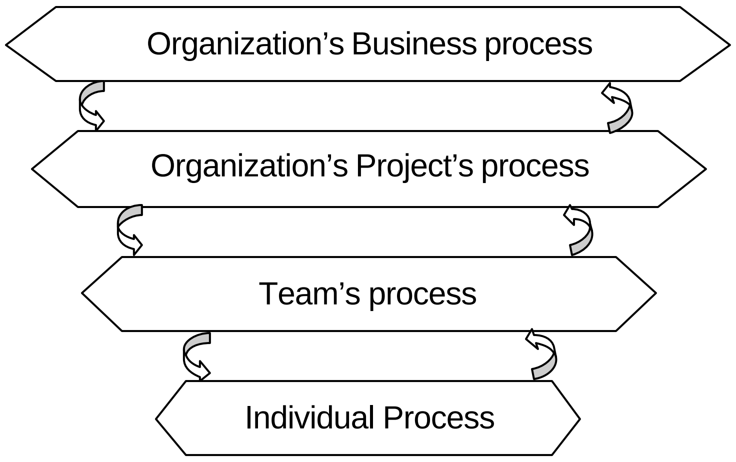

Organization's Business process
Organization's Project's process
Team's process
Individual Process

Figure 4: This figure presents the hierarchy tree of the process layers of a general organization

As you can see from the figure 4, processes on the different levels of the hierarchy are related to each other.

The relationship between the different process levels has two common properties: the lower level process is defined by the higher level process, and achieving the objectives of the lower level process should lead the company to achieve the objectives of the higher level proc ess too.

[Zahran98] also addresses this relationship between the different process levels, by means of highlighting the risk of degrading total business performance of an organization, if there exists a weak relationship between the different process levels in an organization.

To avoid this risk, an organization should establish process focus at a number of levels: the whole business organization, projects and teams, and individual level.

According to [Zahran98], the process focus at the business level should align and streamline the process goals with the business goals. The process focus on the project and team level should define the common practices in support of the project and the team goals. The process focus at the personal level, in its turn, should define working processes and disciplines of individuals, together with actions to improve personal performance. The importance of a relationship between processes at different levels is stressed by [SEI99] also.

Below we describe each level of the process hierarchy tree in more detail:

2.2.1. Organization's business process

An organization's business process defines the organization's strategy, standards, functions and actions that have to be taken in order to realize the organization's business vision and business objectives.

The organization's business process contains number of sub-processes, which concentrate on describing activities that should be done in order to reach the given sub-process goals. Each sub-process goal should be aligned with the goals of the organization's business process, meaning that executing each of the sub-processes should result in reaching goals of the organization's business process too.

[Hunt96] in his book "Process mapping techniques" presents the following list of business processes/sub-process examples:

- Marketing and sales
- Formal strategic and tactical planning
- Budgeting
- Facilities management
- Product/service development and introduction
- Customer service

2.2.2. Organization's project process

The end result of many organizations' business processes is to produce some kind of product or service. To do so, organizations are running different kinds of projects.

[Nicholas01] in his book "Project management for business and technology" defines a project as "a process of working to achieve a goal". According to [Nicholas01], a project's process defines tasks, and organization of the people that are going to perform identified tasks.

Each project's process should be in line with the organization's overall business process, and should ensure achievement of organization's business goals.

2.2.3. Team process

A team process describes the way in which a team is conducting the work. The team process may describe roles, responsibilities, organization, communication routines and so on. The team process is usually a part of the project's process and because of this it is very important that goals of the team process are mapped with goals of the project's process.

2.2.4. Individual process

The individual process describes how a person is conducting his or her everyday work activities. Since teams, projects and organizations are built on individuals, it is not difficult to realize that the employee's individual process can influence the team process and the project's process also. One way to make sure that organization's business objectives are met can be to ensure that the individual processes of the employees are aligned with the organization's overall business process.

2.3. Process diversity in software organizations

How can processes that a software organization is living through be categorized? How can the hierarchy tree of processes, presented in the previous section, fit an organization, which is creating software products? Can we talk about the process hierarchy in the case of software organizations?

These are the questions that we will try to address in this section of our master thesis. To do so, we will try to find out how we can describe different levels of the process diversity tree presented in the previous section, in the light of software organizations.

One can have a suspicion that there exists a diversity of processes in software world, since there exists a waste number of terms that can be used to describe processes that happen in software organizations.

Most of you have probably heard those terms as they follow: Organization's software process, Project's software process, Software development process, Software testing process, Software maintenance process, Personal software process and so on. And again we are getting back to our original question: How do those different software processes relate to each other? Is there some kind of rule that unites all these processes and makes them work for the benefit of the organization?

The figure below describes the connection between the different levels of software processes in the software organizations. The reason for organizing processes in this particular way is to make sure that the activities of individuals, teams and projects within the software organization are contributing to the enterprise business goals and vision.

Each of these levels will be described in more detail in the following sections.

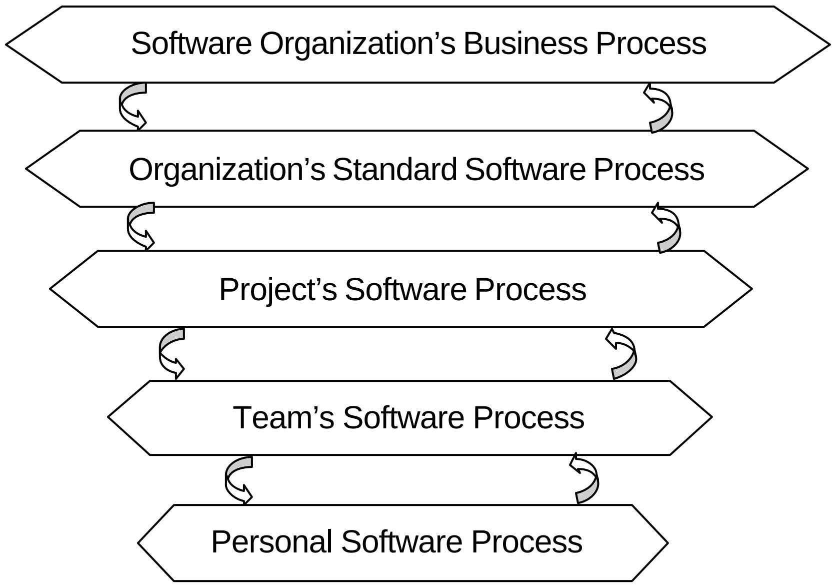

Software Organization's Business Process
Organization's Standard Software Process
Project's Software Process
Team's Software Process
Personal Software Process

Figure 5: This figure presents the hierarchy tree of the process layers of a software organization

2.3.1. Software organization's business process

A software organization's business process can be viewed as a set of activities that enables the organization to produce and sell their services to the customers.

If you will look in the previous section and compare the definition of the software organization's business process with the general definition of the business process presented there, you will not find lots of differences. This is because the software organization's business process is just a subset of all possible business processes. The difference though is that the software organization's business process is concentrated to produce and sell software products and services.

Each software organization's business process will be different, depending on the organization's goals and vision and the mechanisms that are involved to ensure the accomplishment or the identified organizational goals.

Despite these differences, each software company's business process will more or less contain similar sub-processes as marketing, sales, distribution, billing, order processing, customer service, product/service development and so forth.

2.3.2. Software organization's software process

A software organization's software process is just a sub-process of the organization's business process that is concerned with software product development.

According to [SEI99] the organization's software process is a standard process that describes the organization's engineering as well as the management practices and integrates them into the coherent whole.

The organization's standard software process includes standards and procedures that should be followed while developing the software product. It also describes the interfaces, inputs and outputs between different phases of the software process (such as software requirements analysis, software design, implementation, software testing, software configuration management and software quality assurance), which can be considered as sub-processes and can be described separately in more detail.

2.3.3. Project's software process

A project's software process is a way in which the project members should work in order to reach the project goal.

In our case since we are talking about software organizations, the project goal in most cases would be to produce a software product or service. In most of the cases the definition of the project's software process is quite similar to the organization's standard software process, since their goal is the same: to define the way of executing the project, that will lead in successful execution of the project goals.

According to [SEI99] the project's software process is a version of the organization's standard software process, with small changes in it in order to meet the special needs of each project. "The project's defined software process is tailored from the organization's standard software process to address the specific characteristics of the project" [SEI99].

2.3.4. Team's software process

A team's software process describes the roles, responsibilities and communication routines used by the team to reach the common goal. The only difference between the team processes in general and the team's software process within a software organizations is the goal of the team in a software project, which is to develop a software product.

2.3.5. Individual software process

In 1996 Humphrey came up with his book "Introduction to the Personal Software Process" [Humphrey96], where he advocates the importance of the software process at a personal level, and its effect on the organization's performance.

The Individual software process is a process that is performed every day by the software engineer. [Humphrey96] and [Zahran98] believe that by improving the personal software process, one can have a good influence on the overall organization's software process.

2.4. Three essential process factors

In his book "Software process improvement" [Zahran98] argues that most of the existing process definitions do not emphasize three main aspects, necessary to present the essence of the process. Those aspects are: Process definition, Process learning and Process results.

2.4.1. Process definition

The first aspect of the process called "Process definition" is related to the fact that the process should be defined in some way. In other words there should exist a process definition. The process definition can be presented in a document (paper or electronic), which specifies the activities and procedures of the process.

2.4.2. Process learning

The second aspect of the process is related to the fact that after the process has been defined and described in a document, it should be introduced to the people who are going to use and participate in this process. According to [Zahran98], "process knowledge must pass to those who will perform it. The process knowledge should drive the behaviors and activities of those performing the process." This aspect is called process learning.

2.4.3. Process result

The third aspect of the process is the process result itself. The process result can be manifested in products or services produced after executing of the referred process.

2.4.4. Importance of the process factors

The presented factors of the process are important since they are connected with the initial goal of having established the process itself.

To demonstrate the importance of the process aspects, we can look at a simple process called "Prepare a cup of coffee".

In order to reach the goal, which is to get a cup of coffee, a person has to execute a sequence of actions. The actions, as well as their consequence may vary in different cases. In our case actions are as follows:

1. Take a bag of coffee which is stored at place A
2. Pour the measured amount of the coffee in container B
3. Push button C to fill the coffee machine with water
4. Push button "Start" to start the coffee machine

If a person is not acquainted with the process, then it is a definite chance that he will not succeed to prepare the coffee. For example a person might not find the coffee bag, or a person might fail to fill the machine with water.

This demonstrates the need for the process definition and process learning. There should be a document that describes how to prepare coffee, and a person has to have knowledge about how this process works, which requires process learning.

The third aspect of the process called "process results" is important also. By examining the results of the process we can assess how well the process worked and if we have to make changes to the process or not. If by executing of the current process we do not get the right quality of the product, we might examine the process to find the ways of reaching it.

Getting back to our example with coffee, if after performing a process as it was described in the process definition, we will discover that the product (coffee) does not have the right quality; we might get back and make correction in the process definition itself.

For instance if we notice that after executing the process as it was described, we did not get a desirable result, a cup of coffee with sugar, we should make changes in the process description so that it facilitates to get a coffee with sugar if necessary.

This kind of action in the literature is referred as "Process improvement". The improved process description for preparing a coffee will look like this:

1. Take a bag of coffee which is stored at place A
2. Pour the measured amount of the coffee in container B
3. Push button C to fill the coffee machine with water
4. Push button D to add the sugar
5. Push button "Start" to start the coffee machine

One can suspect that in comparison with such a simple process as "preparing a cup of coffee", more complicated processes as building a house or creating a software product are increasingly dependant on the aspects of the process, namely process description, process learning and process results.

2.5. Chapter summary

In this chapter we have described the importance of the three characteristics of the process: Process definition, Process learning and Process result.

We have also looked at the different levels of the software processes in the software organizations. We have concluded that there should be a connection, so called hierarchy tree between different levels of the software processes to make sure that overall goals of the higher level processes are aligned with the goals of low level processes.

Chapter No 3. Software Process Maturity

This chapter aims to present the characteristics of the mature software process, as they are described in the literature and the scientific papers written by different authors.

In the first part of the chapter we compare mature against immature software organizations. The second part of the chapter describes the features of the mature software process.

3.1. Mature Vs. Immature Software Organizations

Depending on how much a company or its management realizes the importance of a process that is producing a product; companies are divided into process oriented and product oriented organizations. In the literature process oriented organizations are often called mature organizations and product oriented organizations are called immature organizations.

In general the difference between product and process orientation is that product oriented organizations are merely interested in the result, whereas process oriented organizations are interested firstly in the process that is producing the result and secondly in the result.

There exists an extensive body of literature describing the features of the mature and immature organizations. In the following sections we have summarized what different authors say about mature and immature organizations and why they think that mature organizations have better chance to reach high quality of their products.

3.1.1. Product Oriented (Immature) Organizations

In the product oriented organizations there is no defined common process to achieve the organization’s goals. The process that is used to produce the result is not important as long as the product is produced.

[SEI99] describes organizations with immature processes as follows:

“In an immature software organization, software processes are generally improvised by practitioners and their managers during the course of the project. The immature software organization is reactionary, and its managers are usually focused on solving immediate problems.”

Since there are no common practices established, to run and control the projects, it becomes extremely difficult to track the progress and keep the initial schedule and project budget. Also there is no common procedure to deal with the critical situations, which increases the risk of project delay.

Even though organizations with immature processes frequently develop product that work, they often exceed the budget and schedule [SEI99]. This opinion is supported by [Paulk et al.93], where the authors name the inexistence of realistic estimates as a reason for exceeding budget and schedule in the projects. Under the time pressure in order to meet the deadlines, projects have to take shortcuts, thereby compromising the entire project quality.

[Humphrey88] compares the immature organizations with the immature drivers, who in case of hurry do not take time to consult the map, but continue driving, thus risking taking a wrong or longer way and waste time and fuel. On the other hand, the mature driver regardless of the hurry will pause to consult the map. The difference between the immature and the mature driver is that later knows the tradeoff between speed and progress.

3.1.2. Process Oriented (Mature) Organizations

An organization with a process focus intends to reach the quality in its products by improving the quality of its process.

A mature organization has already defined a common process that is used to create the product. After the process is defined, it is possible to improve it and thus reach higher quality of the product.

[SEI99] describes mature organizations as follows:

“A mature software organization possesses an organizational ability for managing software development and maintenance processes. It accurately communicates the software process both to existing staff and new employees and carries out work activities according to the planned process”

The quality of the organization’s products is thought to be greatly dependant on the quality of the organizational process itself. In such an organization processes have the characteristics of the mature process: processes are defined, employees are trained in how the process works, and the defined process is followed and monitored.

According to [Paulk et al.93], an important feature of the mature organization is that managers monitor the quality of products and the process that produces them. Well-defined quality goals and quantitative data from the projects make it possible to judge the product’s quality. Having a defined process also facilitates to collect and analyze data from the previous projects, which in turn makes it possible to come up with more realistic estimations of project schedule and budget.

3.2. Characteristics of the mature software process

We have reviewed the works of the different authors who write about the subject of software process maturity, with the goal to come up with the characteristics of the mature software process.

Below we present the list of the characteristics that, according to our literature survey, a mature software process should have:

?? Process contains the basic project control and management practices

?? Process is defined and documented

?? Process is measured

?? Process is optimized

The presented characteristics are supported by the following literature sources: [SEI99], [Humphrey90], [Paulk et al.93], [Zahran98] and [Bennatan92].

In the following sections each of the identified characteristics of the mature software process will be described in detail.

3.2.1. Process contains the basic project control and management practices

One of the basic characteristics of the mature process is that there exist means to manage and control the project. Also practices connected with the requirements analysis and management and the change management are defined and used.

[Humphrey90] and [Bennatan92] in one voice state that without implementing the basic project control and management practices such as requirements analysis, project planning and follow up on the project, the software organization is likely to experience problems such as fire fighting, missed deadlines, budget overruns and producing the products with the low quality.

On the other hand, when the software organization has established management controls, the management can monitor the performance of the project and take actions to avoid critical situations. Project commitments, such as delivery date and budget estimations are more precise, because they are based on the experience from previous projects. The risk of delivering a product, which will have wrong functionality, is minimized due to requirements analysis and management practices. Change management makes it possible to control frequent changes in the requirements and implement them in the orderly way.

3.2.2. Process is defined and documented

This requirement of a mature process refers to the fact that the software organization should have a definition of its software process. In other words, there should exist a defined and documented software process, which will incorporate both the engineering and the management activities of the organization into the coherent whole.

As discussed earlier in chapter 2, an organization can have different levels of software processes. Consequently a company can have several definitions for software processes, one definition for each process level. In this section we will concentrate on the definitions for the project's software process, and the organization's standard software process.

Project's software process is defined and documented

The definition for the project's software process should serve as a guide for the project managers as well as for the software engineers throughout the life cycle of the project.

According to [SEI99], the project's defined process creates a ground for developing the project plan, which describes in detail how the activities in the project will be implemented and managed. The project management uses this information to track the performance and resource costs in order to spot critical situations and react on the problems in time. Software engineers in turn use the project's defined software process to coordinate their work and to make sure that they are using appropriate engineering practices to accomplish the project's tasks.

Organization's standard software process is defined and documented

After a company has reached the point when every software project is executed and managed by means of using its defined software process, a company can extend its maturity level by establishing the defined software process that will be valid for the entire organization.

This means that all projects that will be running in the organization will be compliant with the organization's defined software process, which represents a standard for the processes that are used in each separate project. According to [Paulk et al.93], this standard software process is considered to be a typical process for developing and maintaining software, and should include descriptions for both software engineering and management processes.

After the software organization's standard software process has been defined and documented, each project that will be initiated by the company should base its specific software process definition on the organization's standard software process. [Paulk et al.93] calls this operation "tailoring" and describes it as follows: "Project teams tailor an organization's standard software process to develop their own defined process, which takes into account the project's unique characteristics".

The organization's standard process provides the means to control that different software projects remain in the main qualitative objectives of the company and thus serve the company's overall business goals.

According to [SEI99], another benefit of defining the organization's standard process is that it enables data and experience sharing between the projects. Since all projects are tailored from the same process, the chances that lessons learned in one project will be valid for the others also will increase.

3.2.3. Process is measured

This requirement of the mature process aims to make sure that an organization has an ability to check the productivity of the process that they are using.

According to [Humphrey90], as powerful as the defined process is, it's still only qualitative. The defined process itself does not provide data about how much the process is accomplished, or how effective the process is. Process needs to be monitored and controlled in order to make sure that the process is effective, and is aligned with the business goals of the organization.

The need of process monitoring is also emphasized by [Zahran98]: "If monitoring of the process is absent there will be no guarantee that the process is effective". Without monitoring, the likelihood is that the process will lose the alignment with business and management goals.

In order to be able to monitor the performance of the process that the company is using, the company should identify a set of measurement units. These measurement units should be applicable for the variables of the process, and should give valid data to build a picture of the process efficiency.

According to [SEI99], in order to measure the organization's software process performance, the organization has to establish performance and quality goals for the process. After the goals are in place, the company can measure the performance of the projects' software processes to make sure that they are in the acceptable range.

It goes without saying that having an organization's standard software process will be a great help for a measurement program. Since the organization's standard software process provides a stable ground for the software processes that are used in the different projects, it makes it possible to define the set of measurement variables on organization's standard software process, and use those variables to measure the performance of individual projects running at the company.

Using the same sets of variables to measure the performance for different projects creates a good basis for later data analysis. [Humphrey90] emphasizes the importance of the process definition in order to succeed with the measurements. He claims that the lack of the process definition creates confusion about the specific items that need to be measured. On the opposite, with the defined process one can focus the measurements on specific tasks. The defined process, because of that is an essential prerequisite for the effective measurement.

3.2.4. Process is optimized

This requirement stands for the capability of an organization to analyze the data that was collected after measurements in order to tune the organization's software process itself. Organizations need to assess their current software process in order to decide how well the process works and how well it is aligned with the organization's business goals.

As time goes, the process that was initially defined and worked well in the organization can grow old and become unsuitable for the organization anymore. Software organizations should be able to notice this situation as soon as possible and make the necessary changes to the process. The software organization needs a control on the performance of the process to notice when its process is not beneficial anymore.

In order to stay competitive, software companies need to try out new technologies and new methods. This requires changes to the process also. It goes without saying that after the initial software process has been updated, it has to be measured and analyzed to see if the company has benefited from the changes or not.

[Humphrey90] calls this focus on the improvements of software process itself the “Paradigm shift”. He claims that in many software organizations, analyses of the data, which is collected from different projects, is directed to product improvement. The paradigm shift happens when companies start to collect and analyze data that will help to improve the organization’s software process performance itself.

According to [Paulk et al.93], when a software company satisfies this requirement of mature process, this means that an organization is focused and has a capability of continuous process improvement. This is because the organization has implemented the mechanisms to collect the data on process effectiveness and uses this data to perform cost-benefit analysis of new technologies and to propose changes to the process.

3.3. Does software maturity increase the quality?

In the previous sections of the thesis we have presented the importance of the software process maturity for the organization’s quality and business goals.

In this section we are going to discuss under what circumstances the effort invested in process maturity can result in a waste.

Software organizations nowadays need to see the maturity of their software processes as an organizational asset that enables them to be competitive at the software business market.

[Emam et al. 96] writes that software process maturity has become the driving force at the software process market. According to this article, military and commercial organizations are using the maturity of the software organization as one of the main criteria in their contract-award decisions. Even more, in certain sectors of European economies, high maturity level is a requirement to get a business assignment. From this, one can conclude that software process maturity promises organizations accomplishment of their business goals, as well as some kind of security at the competitive business market.

Motivated by this prospect or maybe because of the hard market requirements, a number of software companies have tried to launch the software process improvement programs in order to increase their maturity level. Many of them are reporting substantial productivity gain as they achieve higher maturity level of their software process. Telcordia technologies [Pitterman00] and Motorola [Daskalantonakis94] are examples of software companies who have benefited from investing effort in improving their software processes.

Despite this positive evidence of software maturity effects on software product quality, some researchers are still questioning the necessity of investing resources and effort in software process improvement projects, and the stability of their effects.

Some researchers claim that the correlation between the software process maturity and the quality is not yet scientifically proven, thus there is no guarantee that the software organization will experience long lasting quality growth, because of the mature process.

[Emam et al. 96] suggest that software companies have been pressured or required to conform to certain standards that define software maturity level, despite adequate empirical evidence that those standards are adequate or effective. The authors are not neglecting the fact that a substantial number of case studies has shown positive effects of process maturity; their concern though is that the audience might have been blinded since organizations that have not experienced process improvement or have even regressed over time will be reluctant to publish their results, so case studies generally tend to show mostly success stories.

Due to that fact [Emam et al. 96] decided to run an experiment, with the goal to identify a relationship between the software process maturity and software quality. Their conclusion was that even though maturity measures was good predictors of overall organizational and project effectiveness, it was difficult to find a direct correlation with improvements in separate software engineering practices, like for instance, requirements engineering.

The suspicion raised by [Emam et al. 96] can be answered. [SEI99] announces that software maturity framework is not a silver bullet and if a company will skip to implement some of the requirements of the mature process, a company cannot expect to achieve the desirable results of quality.

It is strange that software organizations expect to experience substantial growth in the quality, when they do not fulfill all requirements of the mature software process. Such organizations can be compared to people who use the prescription of their medication in the wrong way.

When a person is prescribed some kind of medication, that person is required to follow a strict procedure or process consisting of a sequence of steps in order to succeed with the medication. Many medicines for instance require disciplined intake times; in other case they can cause harm instead of improvement.

Maturity framework also, when applied in a wrong way, or when organization does not implement some of the requirements of the mature process, cannot guarantee desired quality results [SEI99].

3.4. Chapter summary

In this chapter we have compared the process-focused organizations with the product-focused organizations.

The main difference between the process and the product-focused organizations is that the later does not consider the process, but focuses totally on the result of the process. The process oriented organization on the opposite focuses on improving the process, and by means of the process improvement, improving the quality of the process result.

We do not claim that the result of the process is less important that the process itself, the main conclusion here is that it is important to focus on the process to be able to improve the process result.

This chapter also deals with the characteristics of the mature software process. After reviewing the relevant literature, the following characteristics of the mature software process have been identified:

?? Process contains the basic project control and management practices

?? Process is defined and documented

?? Process is measured

?? Process is optimized

Chapter No 4. Effective Software Process

The focus of this chapter is the effective software process. In the beginning of the chapter we will try to build a bridge between the term "software process maturity" and the term "effective software process". We are planning to do this by means of identifying how the software process maturity is connected to the software process effectiveness. Later on we will concentrate on the effective software process and the features of it.

4.1. Connection between mature and effective software process

Process effectiveness can be defined in terms of how well the execution of the process helps the company in realizing its strategic and business goals. In other words, software process effectiveness can be measured by software process performance.

Software process performance is defined in [SEI99] as follows: "Software process performance represents the actual results achieved by following a software process".

It is no doubt that process maturity, discussed in the previous chapter is connected to the figures of the process effectiveness. [SEI99] predicts that as the company moves to the higher levels of the process maturity, software process performance increases too. [SEI99] suggests that with a well-defined process, performance improves in level 3[i] organizations; performance figures continue to increase because of quantitative control in level 4 organizations; in organizations at maturity level 5, process performance is expected to improve continuously.

To demonstrate the connection between the organization's goals, process maturity, process performance and the process effectiveness we have decided to use the following picture:

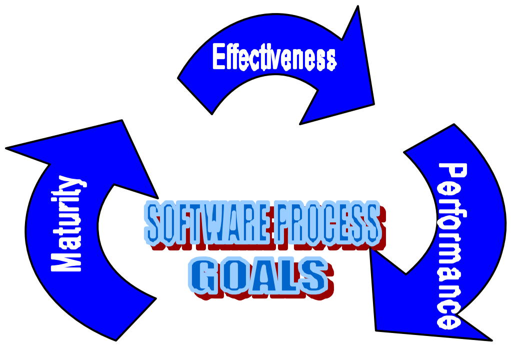


Figure 6: Describes a connection between Software Process Maturity, Effectiveness and Performance.

Since an effective process seems to be one of the crucial aspects of the organization's success, we intend to look closely at the features of the effective software process.

4.2. Features of the effective software process

As in the case when we were looking for the characteristics of the mature software process, we have turned to the available literature to find the features of the effective software process. We have found that among the studied literature, [Zahran98] has described the features of the effective software process in the most explicit way. After this discovery we concentrated to find out if other authors supported statements presented by [Zahran98]. We have found that [Curtis98], [Paulk et al.93], [SEI99], [Humphrey90], [Svensson et al.98] and [Hunt96] have a similar view on the properties of the effective process.

The features of the effective software process, identified as a result of reviewing above listed literature sources are:

?? Following a process is a discipline

?? Process monitoring and ownership

?? Feedback from the process users

?? Process is aligned with the business goals of the software organization.

In the next sections each of these features of an effective software process will be discussed in more detail.

4.2.1. Following a process is a discipline

This feature of the effective process stands for the requirement that employees in the organization should be performing their activities according to the defined process procedures. In the organization where process discipline is established, following the defined process is the norm. The process is followed naturally, since people accept that it helps them to achieve their goals.

To be able to achieve a process discipline in the organization, process understanding, motivation to follow the process and by all means process training is needed. All this can be reached if there exist the process enforcement mechanisms that make sure that the process is understood and followed. [Zahran98] refers to the "Process institutionalization", as a means to establish process discipline in the company.

Process institutionalization means that following a process is supported and encouraged by organizational policies and procedures and supported by the management. In other words following a process is an essential part of organization's culture.

[Curtis98] defines an organization's culture as a system of beliefs and common practices shared among a group of people. Because organization's culture is based on core values that guide people's behavior, it is very important that the organization's software process is institutionalized in the organization's culture.

The author also claims that organizations that use a process standard as a guide for developing a professional software engineering culture, will more likely achieve sustainable gains in the performance. He also underlines that strong organizational culture is important support for new software engineers and managers in developing both skill and faith in existing processes.

[Paulk et al.93] connects process institutionalization to the process maturity and claims that, when a software organization tries to gain maturity, it should institutionalize its software process via policies, standards and organizational structure. Institutionalization means building an infrastructure and a corporate culture that supports the methods, practices and procedures of the process so that they endure after the one who originally defined them has left the company.

4.2.2. Process monitoring and process ownership

This feature of the effective software process is connected to the characteristic of the mature software process “Process is measured”, described in chapter 3.

According to [Humphrey90] processes need to be monitored and controlled in order to make sure that the process is effective and aligned with the business goals of the organization.

If a company does not have data about the performance of the process, company management does not know if the process is helping them to achieve their objectives or not. This may result in the situation when the process that is defined and institutionalized in the company is not beneficial and is a burden rather than a help for business improvement.

Since the need for continuous supervision of the process is obvious, an organization should appoint the persons who will be responsible for this function. The need for this function is highlighted and described in different literature sources. For example [SEI99] advices software companies to appoint a special group of people so called SEPG (Software Engineering Process Group), who will be responsible for the process management function. [Zahran98] refers to people who are in charge of the process management as process owners.

4.2.3. Feedback from the process users

This requirement of the effective software process stands for the fact that management of the software organization should facilitate getting the feedback from the employees that are using the organization’s defined software process. Since designing a software process is a complicated task, it is quite possible that the process definition will need an improvement.

As discussed earlier in section 3.2.4, changes to the process are imposed after analyzing process performance against the organization’s overall quality and business objectives. In this stage, while planning changes to the existing process, software management should make an effort to collect feedback from the individuals who have used the process, since they might provide valuable information about process performance, and also suggestions on how to improve the process.

Process feedback is tightly connected with another feature of the effective software process – a process discipline. If the employees find the existing process not useful or beneficial for them, they will not see the point of executing procedures described in the process, and eventually refuse to follow the process.

4.2.4. Process is aligned with the business goals of the software organization.

This requirement stands there because of the need for the software process to be designed so that its execution will reinforce reaching of the organization’s business goals.

The importance for the software process to be aligned with the business goals of the software organization is highlighted by the several authors. For example [Zahran98] describes the process as the main artifact to support business goals and objectives of the company. “The process is the glue that ties together people, technology, organizational structures and management in a coherent whole, focusing on the business goals and the objectives”.

[Hunt96] is mentioning that an organization’s process should be tightly linked with the organization’s business objectives. “Your business enterprise is as effective as its processes. Business goals can be achieved by only through development of logical business process”.

[Hunt96] also suggests that for reaching the company's business objectives, defining its business process and mapping them back to the original company objectives is crucial. "Process goals are linked both to business enterprise goals and customer's requirements; each process should be measured against process goals that reflect the contribution that the process is expected to make to one or more business enterprise goals".

4.3. Opposite of the effective software process

When can a process be ineffective? One can assume that the process becomes ineffective when it lacks some of the features that have been discussed in the previous section.

If an effective software process is defined as a process, which ensures that company will realize it's strategic and business goals, then an ineffective software process can be described as the opposite.

In order to identify the features of an ineffective software process we have conducted a search in the relevant literature to identify how other authors describe a situation when a software process becomes ineffective. We have identified two scenarios where different authors agree that a process can become ineffective:

?? Software organization's process is not aligned with its business and quality goals

This scenario is supported by [Hunt96], [Zahran98] and [Humphrey90]. It is easy to notice that the given scenario is an opposite of the feature of the effective software process discussed in section 4.2.4.

?? Software organization's process is not followed by software organization's employees

This scenario is described by [Curtis98], and [Zahran98], and is an opposite of a feature of the effective software process described in the section 4.2.1

Below we will try to tell more about presented scenarios, when the software process can become ineffective:

4.3.1. Software organization's process is not aligned with the business and quality goals of the software organization

In this situation the process can be ineffective if its execution does not provide the expected results. This can happen when the process was initially defined in a way that it is not compliant with the organization's infrastructure or its business goals, or a process was initially well defined but with the time, because of the lack of the process ownership and management it became outdated and not aligned with organizations business goals any more. The missing components for effective software process in this scenario would be lack of process monitoring and process ownership.

4.3.2. Software organization's process is not followed by software organization's employees

This situation is an opposite of the institutionalized software process, described in section 4.2.1 and happens when the employees do not follow the defined process.

A process that is not followed can result in not coordinated product development; even more, there can be no guarantee that the process that is practiced instead of the organization's defined software process will be at all aligned with the organization's policy for quality or organizations business objectives.

According to [Zahran98], if a process is not understood and followed, it becomes just a document that can be regarded as a "shelfware" and there is no guarantee that such a process is effective. Software companies are making a mistake, when they think that they have achieved an effective software process, if they have defined and documented the software process that projects are supposed to follow. Unfortunately the mere existence of the document containing the definition of procedures and standards is no evidence that the process is effective. If a process is not understood and followed, a process becomes just a document that can be regarded as a shelfware.

This opinion is also supported by [Curtis98], according to whom, the benefits of implementing a software standard may be illusive if the organization's employees are allowed to undermine practices included in the defined software process. The author also claims that mandating quality practices is often not sufficient to provide quality results. Improved process must be nourished in an organizational culture that perceives the process as a logical implementation of the organization's professional values.

Different authors have listed a number of factors that can result in the software process becoming a shelfware. Below we present those factors in more detail:

Lack of process understanding/Motivation to follow the process

According to [Pressman Nov/ 96] and [Humphrey97] the lack of the process understanding, and motivation to follow the process can be one of the obvious reasons that results in the gap between the defined and the practiced software processes in the organization.

[Humphrey97] highlights the importance to motivate the technical people to execute the tasks that management or the defined software process is requiring from them. If a software engineer does not see or does not understand the purpose of the activity or the procedure that is prescribed by the process it is most likely that he/she will not be motivated to execute them.

[Pressman Nov/96] names software process practitioners perceptions to be one of the main obstacles that has to be overcome in order to succeed with software processes.

It is mentioned by a number of authors that software engineers are often reluctant towards the disciplined process, or the process improvement activities that require following formal procedures, like writing and updating design specifications, conducting software inspections, or reviews and so on. For example [Gilb et al. 93] write that in the software company there exists a default resistance to the change that is caused by introducing in the process such a formal method as software inspections.

The reason for these misbelieves is based on the fact that the formal methods or procedures are more or less time consuming and time appears to be one of the critical resources in software development. According to [Gilb et al. 93], people in software organizations perceive the software inspections as a time consuming activity because they feel that it takes much more time to check a document than they have available. However, the time limit is not the only reason that results in practitioner's doubts in the disciplined process. The lack of understanding what are the benefits of following the defined software process may as well play the crucial role for a process to be accepted by the practitioners. [Gilb et al. 93] explicitly show that one of the reasons why the software inspections may be ineffective in a company is that management has failed to make visible what are the benefits of executing software inspections.

[Pressman Nov/96] informs us that many software engineers perceive software process improvement and quality assurance activities as a bottleneck to rapid process, because they do not know how it will help them in performing their everyday activities. The author suggests that management should put an effort in changing these perceptions by providing answers to the practitioner's question "What's in it for me?" [Pressman Nov/96] suggests that the education programs and training are common ways to widen the understanding of software engineers.

Lack of process training/Education

Software engineers have to be familiar with the procedures of the software process to be able to follow it. If software engineers do not receive the information about the software process and necessary training to follow it, then it is a high chance that they will not be able to execute the process just because of the lack of information. For example if peer reviews are introduced first time into the process, employees have to be aware about this change firstly, and secondly they have to get a proper training to in order to be able to perform peer reviews on the resources.

Many authors support the importance of process training. [Zahran98], [Humphrey] and [SEI99] agree that without the proper software process training, it is doubtful that a company will achieve an effective soft ware process.

[Pressman Sep/96] writes that many software companies have difficulties to achieve the mature and thus effective software process because of the lack of knowledge among the software developers and the managers. He reports the results of the Software Engineering Self Assessment Test (SESAT) where 2,600 software engineers have tested their knowledge in the software engineering discipline. Test was focused on 10 critical management and technical areas, such as Business issues, Software project management, Software quality assurance, Software configuration management, Software metrics, Analysis methods, Software design and coding, Testing methods, and Software maintenance and reengineering.

The outcome of SESAT was that software developers showed relatively strong knowledge in coding and programming languages. The results in configuration management, software metrics, software analysis, design and testing were noticeably low. These results show the necessity of software process training for the organizations that aim for high maturity levels.

Lack of process feedback

According to [Zahran98] absence of the software process practitioner's feedback on the process can lead to the ineffective software process. [Svensson et al.98] reinforce this idea by stating that when management acknowledges the feedback of the employees, the employees feel that they are part of the process engineering, which increases their motivation.

When management in a software organization decides to tune the software process performance, it is very important that they take into consideration the employees' opinions about the process performance and if the process was beneficial for their working assignments or not. In the case when the feedback from the process users is overlooked, there is a risk that the process will become a shelfware.

4.4. Chapter summary

In chapter 4 we have identified the connection between the mature and the effective software process. The connection is as follows: the higher is the software organization's software process maturity; the better is the chance that the software process of the organization will be effective.

We have conducted a study of the relevant literature sources to find the characteristics of the effective software process. Identified features of the effective process are:

?? Following a process is a discipline

?? Process monitoring and ownership

?? Feedback from the process users

?? Process is aligned with the business goals of the software organization

We have also identified the scenarios when a software process can be ineffective. According to scenario 1, a software process is ineffective when it is not aligned with the business and quality goals of the organization.

Scenario 2 happens when the employees do not follow the defined process of the software organization and the process turns into the "shelfware". We have concluded that there are number of factors that can cause scenario 2. Those factors are:

?? Lack of process understanding

?? Lack of motivation to follow the process

?? Lack of process education and training

?? Lack of process feedback

[i] CMM describes the maturity of software process in five different maturity levels. The higher is the level number the higher is the maturity of the software process.

Chapter No 5. Case Study

The empirical part of the research was conducted at one of the functional departments of a large telecom company in Scandinavia. To keep the company anonymous, the department will be referred to as “ABC”. The department is responsible for developing and maintaining the family of software products that we in this thesis refer to as the product.

Company management was interested in the study since they suspected that they were missing some information about their software processes that was practiced in the projects.

The management of “ABC” was interested to receive an answer on the question if there was a difference between their standard and practiced software processes, and to identify the reasons that caused the difference.

Company management also expected to receive a process map for the software process practiced in their projects, a list of potential problems in the process and the suggestions for the improvements.

5.1. Case study objectives

Below the reader will find the list of the objectives that were identified for the case study. These objectives are:

?? To identity if there is a difference between the defined (standard) and practiced project’s software process in the department

?? To identify the possible causes for the difference between the processes

?? To collect the software process practitioner’s feedback on the practiced process in order to identify potential problems in the process

?? To suggest process improvement activities for the department

?? To create the software process map, describing the practiced software process at the department.

This case study can be identified in the boundaries of our research, since it will help to realize one of the main research goals presented in section 1.3, namely to find the factors that can negativity affect the process effectiveness at a real company.

For instance by means of finding differences between the defined and the practiced software processes at the company, and possible causes for it, we plan to find at least one reason for a process at a real company to be ineffective. (As discussed in section 4.3.2 the difference between the defined and the practiced software processes is one of the main characteristics of an ineffective process)

The feedback from the employees can help to identify other factors, than the difference between the defined and the practiced processes, which could also affect the process efficiency in a negative way.

The last goal is a requirement from the company’s management. It is not directly connected to the main study goals but company management believed that the process map would help them to see the differences between the defined and the practiced software processes. For more information about the process map please see the appendix.

5.2. The case study method

The method that will be used to reach the case study objectives is based on the interviews and the study of the company documentation.

5.2.1. Document study

The aim with the document study is to get familiar with the documentation that describes the project's standard software process at the department. Without this study it would not be possible to see the differences between the standard and the practiced software processes.

We have studied the company's organizational structure and their procedures for documentation handling to find the necessary documentation. The list below presents the documents that were used to study the project's defined software process at the department.

Process descriptions found at the company:

?? "ABC" Development process

?? "ABC" Test process

Other documents:

?? "ABC" Milestone criteria

?? Project specification for "ABC" development project

?? Main requirements specification

5.2.2. Interviews

The interviews with employees at the department are used as the main tool to collect information about the software process that was practiced in the projects. In general, data gathered from the interviews will be used to find out:

?? The description of the software process that is practiced by the employees

?? The degree of software process awareness among the employees

?? Employee's feedback on the software process that they are using

5.2.2.1. Design of the interview questionnaire

After the nature of the data that had to be collected was decided, the questionnaire, which would be used throughout the interviews, was created. The questionnaire was constructed having the following points in mind:

?? Use plain language

?? Open-ended questions

The example of the questionnaire can be found in the appendix.

Here should be admitted that despite the efforts to keep the questionnaire unchanged in all interviews, in some cases depending on the interviewee position and knowledge area, we had to ask some additional questions. This was also caused by the fact that as we moved on with interviews, our knowledge about the processes at the company grew, which resulted in some modifications of the original questionnaire.

Anyway, we want to acknowledge that to maintain the interview material structured and to save the questionnaire logics, we had a rule not to modify a question without checking that the logic of the questionnaire was maintained.

5.2.2.2. Choice of those interviewed

After reviewing the project organization pattern at the department, we decided that in order to get the wider view of the software process at the department and also to have a possibility to cross-check the information, we should follow the organization tree and interview people starting from the project managers ending with the designers.

In the beginning of our planning we thought we would interview employees that were involved only in the product development phase, but as we moved on with the interviews we found it necessary to interview people from the test department, the system management and the configuration management.

A table below is showing the number of the people interviewed and their positions at the company:

Project managers	**2**
Technical project managers (technical coordinators)	**2**
System architects	**2**
Designers	**3**
Configuration managers	**3**
System management	**2**
Test managers	**1**
Quality coordinators	**1**
Total:	**16 persons**

5.2.2.3. Interview Procedures

Here are some characteristics for how the interviews were conducted:

?? All interviews were face-to-face meetings

?? Approximate interview time was 1 hour

?? All interviewees were asked the questions from the same interview questionnaire and in some cases additional questions also. (This depends on the nature of the interview.)

?? We have tried to avoid using the tape recorder to avoid stressing the interviewee (Unless it is necessary). In total the 80% of the interviews were done without the tape recorder. (We avoided using of the tape recorder because we noticed that interviewees got a bit nervous if they knew that the interview was recorded).

5.2.2.4. Data verification and correctness

Since the material that we have got after conducting all interviews was the ground for the further analysis and our conclusions, it was crucial that the information that the company employees were giving to us was not changed or somehow corrupted by our misinterpretations.

To ensure data correctness we used the following approach:

Face to face meetings:

All the interviews were face-to-face meetings, so that we could re-ask the questions in case we were not sure about the answer of the interviewee.

Interview summaries:

One risk with how we conducted the interviews was that in most of the cases we were just taking notes as the interviewee was answering our questions. It is a definite risk here that because we were going to make the final analyses of the material after all the interviews would be finished; we could forget what our notes were about and by that make a wrong interpretation of the note.

Another risk is more general, that is a characteristic of interview techniques: since there is a time-pressure during an interview and you want to get an answer on all questions, you can misinterpret the answer of the interviewee during your conversation but not notice it. The best way to help this situation is to let the interviewee read how you interpreted his or her answers. Unfortunately this was not the option for us because the reasons connected with the time-budget of company employees and ours.

Having in mind all that was said, we used the following way to protect data from corruption:

After each interview was finished, the interview summary would be created, which would be based on the notes that we made at the interview. Since this summary was made just right after the interview, we hoped it would lessen the chance of missing information and misinterpretations.

After the interview summary was finished we would review it in order to find out if there was something in the answers that could have been misunderstood by us, or if there was some information missing. In this case we would contact the interviewee for further clarifications.

Cross-checking

Since all our interviews were with the people who were working at the same department, we could cross check the information they gave us against each other. This gave us one more chance to verify the correctness of the information.

5.3. Analysis

5.3.1. Method for the analysis

After all the interviews were done and the interview summaries were created, we put all the material together and tried to identify findings and conclusions.

First thing that we did at this stage was that we went through all interview summaries so that we could get a feeling on how to proceed with our analyses.

Due to the fact that all the questions in the interview questionnaire were open ended, which gave the interviewees freedom to speak, it resulted in the situations when:

?? Interviewee was giving an answer on the other question that he was asked,

?? Interviewee was giving an information on several questions in one answer,

?? Interview has misinterpreted the question and gave the answer that was not in connection to the question.

Taking into consideration all these aspects, we have decided to structure the interview material in the following way:

First step:

We have looked at the questions that have been asked to the interviewees and made sure that all the questions were unique. Unique questions were the questions that were interpreted in the same way by the interviewees.

The question that had several interpretations was divided into the sub-questions, each corresponding to each way of interpreting of the original question.

On the other hand if two questions were interpreted in the same way we have joined these questions into one and have made a new formulation for that joint- question.

Second step:

We have made sure that all answers were tied to the questions that they correspond to.

This was done to help the situation when the interviewee has given an answer to the other question rather than the one that was asked. Or the situation when in one answer the interviewee was answering several questions.

Third step:

We have grouped all the answers to a specific question in the tables. After grouping of the questions was finished, the tables with the answers were used as a main source for the analysis.

5.3.2. Criteria for identifying of the findings

After collected data was structured in the tables, the challenge was to identify which questions and answers to those could be used to form the findings for the case study.

A criterion for forming the findings based on the interview material is as follows:

Criteria 1: Because this is a qualitative study, a single answer on a specific question can form a finding. This is possible to do according to the definition of the qualitative research. As discussed in section 1.4.2, for a qualitative research, an existence of a quality in the studied object is more important then the quantitative characteristics of it.

Criteria 2: Only those answers that are considered to be helpful to make conclusions regarding the main questions of the case study can form a finding.

In the next section the data that was selected to form the findings, as well as findings themselves are presented.

All findings are presented in tables where the reader can see the finding itself, answers from the interview material that formed the finding, so called “ground data” and also explanatory information where necessary.

5.4. Case study findings

The Data that was selected to build on the findings is presented with respect to the objectives of the case study.

✍✍ **Identity if there is a difference between the defined (standard) and the practiced project's software process in the department**

Selection ratio

In order to find the ground data, we had to review the questions from the first section of the questionnaire, to see if we could find any differences between the process described in the project's documentation and the process described by the employees.

Data was selected as a ground for the findings if it described parts of the defined software process that was not included in the standard documentation, or if it indicated that some parts of the standard software process were omitted in practice,

Findings identified:

Having in mind the selection criteria presented above, the following findings have been found:

Finding 1	**Description of the standard software process misses information about it's sub-process called "Building process"**
Ground data	Most of the interviewed employees identified part of the practiced software process, which was not described in the documentation for project's software process. This missing part is tightly connected with the configuration management practices of the project. Employees refer to it as a "Building process".

Note: Activities in the "Building process" are carried out by the software CM. The purpose of those activities is to create the executable package, so called "build of the product" at the end of each increment. More information about this sub-process can be found in the process map, presented in the appendix of this thesis.

Finding 2	**Inspections of DS and the source code do not always take place**
Ground data	According to 2 out of 3 interviewed designers, inspections on the source code and even on the DS, that are mandatory in the documented process, can be omitted in reality

Finding 3	**Writing DS before implementation is not always the requirement**
Ground data	Designers can get started with the implementation phase before they start to work on DS. (2 designers out of 3 interviewed provided this information). This contradicts the documented description of the development process of the company.

One can easily notice that all findings presented above indicate that there is a difference between the defined and the practiced project's software process at the department.

✍ Identify the degree of the software process awareness between the employees

Selection ratio

Questions and answers from the second section of the interview questionnaire were targeted as the potential ground data for the findings. The choice is justified by the fact that questions in that section of the questionnaire was specially designed to address the employee's awareness of the process that he or she is participating in.

Findings identified:

Finding 4	**Most of the interviewed designers have not read the description of the process that they are supposed to follow**
Ground data	2 out of 3 interviewed designers have not read the project's software process description.

Finding 5	**Some designers do not see the meaning and the benefit of the paper work that they are obliged to do according to the process**
Ground data	One of the interviewed designers stated that it was difficult to see the benefit from writing the documents such as design specification, technical report and change impact record.

Finding 6	**The key person in the project is not motivated to facilitate process work**
Ground data	One of the key persons in the project did not believe in the benefits of spending time on process or process improvement efforts.

Note: for the reasons of the confidentiality we do not mention the role of the key person in the project.

✍ Collect the software process practitioner's feedback on the practiced process

Selection ratio

Questions and answers from the third section of the interview questionnaire were used to identify the ground data for the findings about the employee's feedback on the process.

The findings are presented in respect to the part of the project's software process that they belong to:

Finding 7	**There exists the need for better requirements engineering**
Ground data	5 out of 16 employees identified that one of the problems in the project's process was the main requirements specification, which was difficult to interpret. 3 employees identified the need for better change management of the requirements.
Finding 8	**The need for better time estimates**
Ground data	5 employees have noticed that the time estimations were not good, and needed an improvement.

Finding 9	**Procedures connected with "Building process" should be improved**
Ground data	3 employees have underlined that the building procedure did not work well, since the designers did not provide source code that was supposed to be included in a certain build of the system.

Finding 10	**The need for knowledge management**
Ground data	4 employees have identified the need of experienced people at the department. 2 persons have identified the problem that the project was entirely dependant on one person, who had the most knowledge.

5.5. Case study conclusions and the suggestions for the improvements.

In this section we will present the case study conclusions, which are based on the findings listed above. The conclusions will be used to answer the questions: What are the factors that negatively affect the software process effectiveness at the studied software company? And how can the studied organization avoid those factors?

5.5.1. Case study conclusions

Conclusion 1: There is a difference between defined and practiced software processes at the studied software company

This conclusion is based on the findings 1-3. (See section 5.4). These findings explicitly show the difference between the defined and the practiced software process: The practiced software process contains a sub-process that is not described in the defined software process, and software engineers are not always following the mandatory activities described in the defined software process.

This dfference can be regarded as an indicator of the ineffective software process, because it contradicts the attribute of the effective process, called process discipline. This characteristic of the effective software process is described in section 4.1 and stands for the requirement that employees are performing their activities according to the defined process procedures.

Conclusion 2: There are factors that resulted in such a difference

This conclusion was found by means of analyzing the findings 1-3 together with the findings 4-6. In order to explain how the factors that might have caused the difference have been identified; we have to look more closely at the findings.

Lack of the process ownership

Finding 1 shows the difference between the defined and the practiced software processes because it indicates that the defined software process does not describe a sub process (Building) that is widely used by the software engineers in their daily routines.

We have figured that this difference might be caused by the lack of process monitoring and process ownership. Process monitoring and ownership is one of the characteristics of the effective software process as discussed in section 4.1.

We think that the company could have avoided this problem if they would assign some person a responsibility to review the practiced process, and compare it to the standard process definition.

Unfortunately we do not have any hard data from the interviews that proves that the company did not have a responsible for the process, so this will be regarded as just a suggestion.

Following a defined process is not enforced and institutionalized

We have reviewed findings 4-6 to identify the factors that might be affecting the software engineer's behavior towards the defined process.

According to finding 4, most of the interviewed software engineers have not read the description of the process that they are supposed to follow. Most of them have named time pressure and the size of the document that describes the defined process as a reason for it.

Finding 4 gives us a ground to think that there is a lack of process institutionalization mechanisms at the department. For the studied company lack of the process institutionalization means that the company management has to put more effort to make the employees aware of the processes and make sure that the defined software process is followed.

Software engineers are not motivated to follow the process

Findings 2 and 3 show that the difference between the defined and the practiced software processes is due to the fact that software engineers are not following the procedures appointed in the defined software process. A process that is defined but not followed by the practitioners is referred to as a shelf ware in section 4.2. According to the literature study, a process can become shelf ware in case of lack of the process understanding, or lack of motivation to follow the process.

Findings 5 and 6 show that there is a low motivation to follow the defined software process among the software engineers. We can make this conclusion because of the fact that some of them do not see the point of performing mandatory activities connected with the paper work.

We believe that low motivation towards the process work; especially among the key persons of the project can negatively influence motivation of the other project members to follow the defined software process. We also suspect that most of the designers skip mandatory activities such as inspections of DS and the source code because they do not see or do not understand the benefit of it.

Conclusion 3: The feedback from the process-users can help the company to increase the process performance

Findings 7-10 present the software engineer's feedback on the process that they are using. As one can see findings 7-10 present a number of improvement points that the company management should take into consideration. We have found that receiving the feedback on the process is valuable information that can be used to improve the process performance and reach an effective software process.

5.5.2. Suggestions for improvements: How a company can avoid those factors?

In the previous section we have identified the number of the negative factors that has resulted for the studied company to have a difference between the defined and the practiced software processes.

This section presents suggestions for a studied software company on how the factors that negatively affect the effectiveness of the software company can be avoided:

The improvement suggestions are presented in the tables where the reader can find the suggestion itself, a negative factor that the suggestion corresponds to, preconditions of a negative factor and the result of the negative factor.

Negative factor	**Lack of the process ownership and monitoring**
Factor precondition	There is no mechanism to check if there is any difference between the defined and the practiced software process.
Factor effect	The defined process can become outdated; there will be a difference between the defined and the practiced software process.
How to avoid the factor	Appoint the persons or groups of the persons who will be responsible for the process monitoring and making necessary updates to the process.

Negative factor	**Following a defined process is not enforced**
Factor precondition	There is no mechanism to ensure that the defined process, or changes to it are introduced to the practitioners.
Factor effect	Software engineers do not read the process description; they follow their own version of the process.
How to avoid the factor	The management in the software organization should put an effort to show to the employees the importance of the process. Process training activities should be planned and facilitated by the organization's management. Also company management should make it easy for the software engineers to read and understand the description of the software process. According to the interviewed software engineers they were not motivated to read the process description because of the time pressure and the size of the process description document. According to [Hunt96], creation of a process map for the defined software process would help the process practitioners to interpret and understand the contents of the defined process.

Negative factor	**Software engineers are not motivated to follow the process**
Factor precondition1	Software engineers do not understand the purpose or benefit of certain activities included in the defined software process.
Factor precondition2	Key persons in the project are not motivated to facilitate process work, which influences the motivation of the other software engineers also.
Factor effect	Software engineers are not practicing the activities specified in the defined software process; they create their own version of the process.
How to avoid the factor	Design educational programs, where the purpose and the benefit of the activities included in the defined software process will be explained both to the experienced as well as to the new employees of the company.

Chapter No 6. Discussion and Conclusions

In this chapter we will try to summarize our findings from the literature survey and the case study and present the final conclusions of the thesis

6.1. Discussion

The goal of the given section is to search for the similarities between the list of the factors identified in the literature survey as the negative factors for the effectiveness of the process and the negative factors found in the case study.

After reviewing the results of the literature survey and the case study we have concluded that all the negative factors identified as a result of the case study is supported by the literature survey. The tables below present similar negative factors found in the case study and the literature survey:

Negative factor Case study:	Lack of the process ownership.
Negative factor Literature:	Section 4.2.2 in the literature study describes the process ownership as one of the features of the effective software process. This means that the lack of the process ownership can be regarded as a factor that causes a process to be ineffective.
Conclusion:	We consider that the finding from the literature survey supports the finding from the case study.

Negative factor Case study:	Following the process is not enforced.
Negative factor Literature:	Section 4.2.1 states that without process enforcement it is not possible to achieve the process discipline, which is one of the characteristics of the effective software process.
Conclusion:	We conclude that both the case study and the literature survey indicate towards the same negative factor.

Negative factor Case study:	Software engineers are not motivated to follow the process.
Negative factor Literature:	The literature describes the lack of the motivation to follow the process, as one of the factors that cause an ineffective software process.
Conclusion:	Since both the case study finding and the literature survey result describe the same phenomena, we consider that the finding from the literature survey supports the finding from the case study.

It was important to identify similarities between the case study findings and the literature survey performed in this paper, since it will strengthen the value of the case study findings and conclusions.

We hope that by means of identifying similarities between the results of the case study and the opinions from other researchers collected in the literature survey, we will increase the chance that findings discovered during case study at "ABC", will be applicable for some other software companies too.

Below we present a diagram that shows the dependences between the different negative factors identified both in the literature survey and the case study. It also aims to show how those factors influence the effectiveness of the software process.

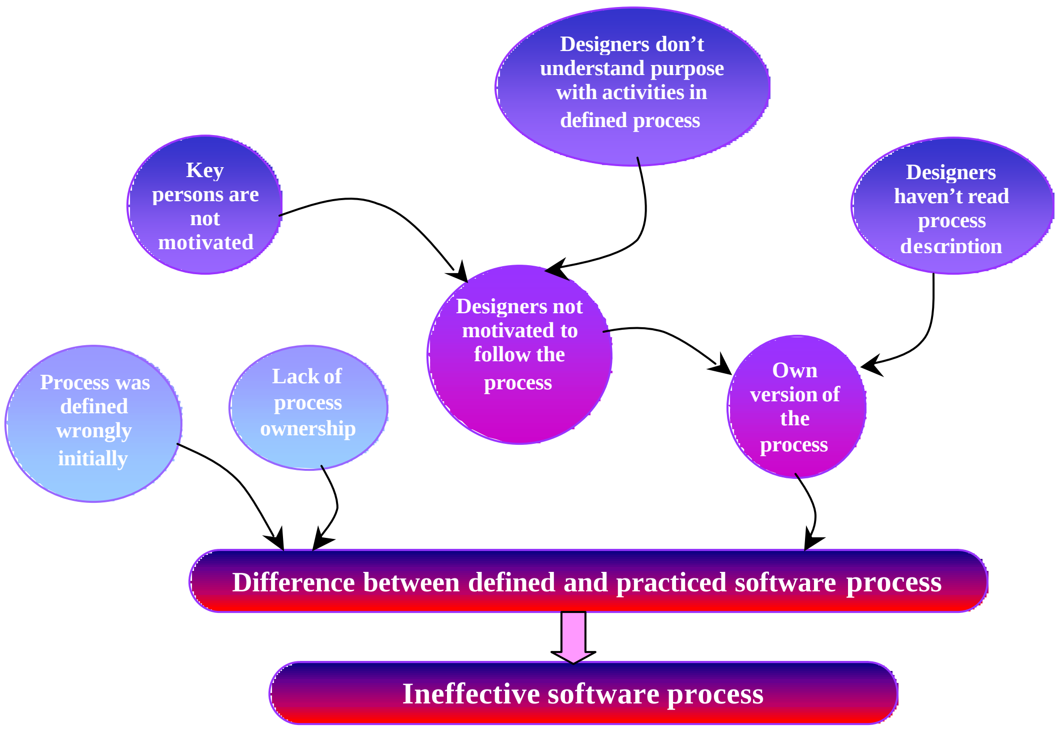

Designers don't understand purpose with activities in defined process
Key persons are not motivated
Designers haven't read process description
Designers not motivated to follow the process
Process was defined wrongly initially
Lack of process ownership
Own version of the process
Difference between defined and practiced software process
Ineffective software process

Figure 7 This picture shows the relationships between the factors that negatively affect the effective software process in the software company.

6.2. General conclusions

This study has been initiated with the intention to help software companies to reach the mature and effective software processes in their organizations. The aim of the study was to identify factors that negatively affect software process effectiveness in software organizations and to suggest the ways to avoid those factors.

To accomplish the aim of the investigation the literature survey and the case study at the real existing software company has been conducted. Below the reader can find what we have concluded as a result of the study.

6.2.1. Different levels of software process

There can be found different levels of software processes in the software organization. Those processes are: software organization's business process, software organization's software process, software organization's project process, software organization's team process, and personal software process.

All these different processes are connected to each other and are supposed to contribute to the business objectives of the software organization. In this paper we have mainly concentrated on the project's software process in a software organization.

6.2.2. Characteristics of the mature software process

After reviewing the relevant literature, the following characteristics of the mature software process have been identified:

?? Process contains the basic project control and management practices

?? Process is defined and documented

?? Process is measured

?? Process is optimized

6.2.3. Characteristics of the effective process

We have conducted a study of the relevant literature sources to find the characteristics of the effective software process. Identified features of the effective process are:

?? Following a process is a discipline

?? Process monitoring and ownership

?? Feedback from the process users

?? Process is aligned with the business goals of the software organization

6.2.4. Negative factors and how to avoid them

Software organization's process is not aligned with the quality and business goals of the organization

This factor can happen when the process is engineered without taking into consideration the organization's quality and business goals. Given factor was identified as a negative factor in the literature study. We did not have a chance to investigate this factor in the case study, since it requires a deep knowledge of the organization's quality and business goals that we did not have time to acquire. Nevertheless we think that software organizations should take great care when they are deciding what their software process will look like.

There is also another reason when an organization's process can become outlined from its overall goals. The organization has to make sure that its process is measured and monitored. Without process monitoring it is difficult to know if the process is beneficial for the company or not. We suggest that companies should assign special persons or groups of persons who will be responsible for carrying out process monitoring activities.

Lack of process ownership

This factor happens when there is no mechanism at the company to identify if there are any differences between the defined and the followed software process in the organization. The existence of the difference between the defined and the followed software processes can be an indication of the problem or the need of improvement.

We suggest that software companies appoint people who will be responsible to identify differences between the standard and the practiced software processes and also possible reasons for the differences.

Following a defined process is not enforced

Software companies have to make sure that the defined software process is actually practiced and followed by the projects and project participants. If a software company has defined software process that requires executing some formal procedures like inspections, peer reviews or something else, a company can not expect to see positive results of those methods if they are not practiced in the projects.

We suggest to the management in software organizations to put serious effort in institutionalizing the defined process into the everyday life of the organization and its projects.

There is a list of the actions that management can do in order to enforce the defined software process into the organization:

Management should make sure that software engineers are familiar with the defined software process. The authors recommend usage of training programs and process maps to facilitate the process learning within the organization.

Management should make sure that practitioners of the software process realize the benefit of performing activities that are specified in the defined process.

Software engineers are not motivated to follow the process

We have concluded that without motivation, software engineers will most probably not follow the procedures of the defined software process. Making the benefits of the defined process procedures obvious to the practitioner is believed to be one way of increasing practitioner's motivation.

6.2.5. Last points

In the end we would like to mention that negative factors that are identified in this thesis are not the only factors that affect the performance of the software process in the companies.

We realize that such activities as choosing a suitable development method or a test technique has a great effect on the software project's performance also.

Our point though is that in order to experience expected results from any method, will it be Pair programming, Cleanroom engineering or any other method, the software company should make sure that the company employees are educated and motivated to execute the selected method.

We believe that this thesis will be a contribution for the software world, because it addresses one of the existing problems in software engineering - how to reach the effective software process. In this study we have tried to identify reasons why a software process can be ineffective and have made suggestions on how to avoid factors that negatively affect the processes in software organizations. If a software organization will take into consideration the existence of the negative factors identified in this paper, there is a chance that it will improve effectiveness of the software process.

Our findings discovered in the case study are similar to the findings of the other authors who have been investigating effectiveness of the software process, so we hope that the outcome of this study will be useful not only for the company where the case study was conducted, but for the other software companies too.

6.3. Suggestions for the future work

As much as we hope that our research has provided answers on some questions, we know that there are still many issues that need to be answered in the future. In this chapter we will present some suggestions for the future investigation.

6.3.1. Suggestion No1

One of the conclusions of our study was that the software process is ineffective when it is not aligned with the software organization's quality and business goals.

We think it would be interesting to investigate how different software organizations' software processes are aligned with their overall goals, identify if there are any weak points and try to construct a model that software companies can use while engineering their software process.

6.3.2. Suggestion No2

One of the deliverables of the case study presented in this paper is a process map, describing how the project's software process works in the studied organization. We have suggested to the management of the company to use the developed process map to increase software engineer's awareness of the software process that should be practiced in the company.

According to [Hunt96], a process map can be very helpful to increase overall process awareness in the company. We also suspect that software engineers will more likely review a process map, rather than read a document describing a defined software process, which according to them is a "Huge document filled with a lot of text".

We think it would be interesting to run an experiment in order to see how using of the software process map as a main reference for the defined project's software process in the company can effect the project participants' awareness of the process.

Chapter No 7. Appendix

7.1. References

[Bennatan92]	E. M. Bennatan **Software Project Management: A practitioner's approach** McGRAW-HILL book company Europe 1992
[Curtis98]	Bill Curtis **Which comes first, the Organization or Its Process?** IEEE software November/December 1998
[Daskalantonakis94]	Michael K. Daskalantonakis **Achieving Higher SEI levels** IEEE software July 1994
[Emam et al. 96]	Khamed El Emam; Nazim H. Madhavji **Does Organizational Maturity Improve Quality?** IEEE software September 1996
[Gilb et al. 93]	Tom Gilb, Dorothy Graham **Software Inspections** Addison-Wesley Longman Limited, 1993
[Hoepfl97]	Marie C. Hoepfl **Choosing Qualitative Research: A Primer for Technology Education Researchers** Journal of Technology Education Volume 9, Number 1 - fall 1997
[Humphrey88]	Watts S. Humphrey **Characterizing the software process: A Maturity Framework** IEEE software March 1988
[Humphrey90]	Watts S. Humphrey **Managing the Software Process** Addison-Wesley Publishing company, Inc. 1990
[Humphrey96]	Watts S. Humphrey **Introduction to the Personal Software Process** Addison-Wesley Publishing company, Inc. 1996

[Humphrey97]	Watts S. Humphrey **Managing Technical People** Addison-Wesley Longman, Inc. 1997
[Hunt96]	Daniel Hunt **Process mapping** **How to Reengineer your Business Process** John Wiley & Sons, Inc.
[Lindvall et al.2000]	Mikael Lindvall, Loana Rus **Process Diversity in Software Development** IEEE Software, July/August 2000 Vol. 17 No. 4
[Nicholas01]	John M. Nicholas **Project Management for Business and Technology** Second Edition Prentice-Hall 2001
[Ost87]	Osterweil, L. **Software processes are software too** 9-th Int. Conf. on Software Engineering, 1987
[Patton90]	Patton M.Q. **Qualitative Evaluation and Research Methods** (2nd ed.) Newbury Park, CA: Sage Publications, Inc. 1990
[Paulk et al.93]	Mark C Paulk, Bill Curtis, Mary Beth Chrissis **The Capability Maturity Model, Version 1.1** IEEE software July 1993
[Pitterman00]	Bill Pitterman **Telcordia Technologies:** **The Journey to High Maturity** IEEE software July/August 2000
[Pressman Nov/96]	Roger Pressman **Software process perceptions** IEEE software November 1996
[Pressman Sep/96]	Roger Pressman **Software process impediment** IEEE software September 1996
[Robson93]	Colin Robson **Real world research** Blackwell Publishers Ltd, 1993

[SEI99]	Software Engineering Institute **The Capability Maturity Model:** **Guidelines for Improving the Software Process** Addison-Wesley Longman 1999
[Strauss et al. 1998]	Anselm Strauss Juliet Corbin **Basics of Qualitative Research** Sage Publications, Inc. 1998
[Svensson et al.98]	Pontus Svensson, Dzagen Milicic **Sparks to a living quality Organization-** **A total quality approach towards improvements** A master thesis in software engineering Department of Computer Science and Business Administration University/College of Karlskrona/Ronneby, 1998
[Zahran98]	Sami Zahran **Software Process Improvement** Practical Guidelines for Business Success Addison-Wesley Longman 1998

7.2. Abbreviations and definitions

7.2.1. Abbreviations

CIR	Change Impact Record
CM	Configuration Management/Configuration Manager
CMM	Capability Maturity Model
DS	Design Specification
IP	Implementation Proposal
MD	Modification Description
MP	Milestone Plan
MRS	Main Requirements Specification
PM	Project Manager
PS	Project Specification
PTP	Project Time Plan
SA	System Architect
SPM	Strategic Product Manager
TI	Technical Investigation
TPM	Technical Project Manager
TR	Technical Report
TRR	Trouble Report

7.2.2. Definitions

Software process	A software process is a set of activities, methods and practices that people employ to develop and maintain software and its associated products [Paulk et al.93].
Software process performance	Software process performance represents the actual results achieved by following a software process [SEI99].
Effective software process	A software process can be considered to be effective, if its result is a product that is compliant to the software organization's business and quality goals.
Software build	A compiled and merged version of the latest source code of the software product.

7.3. Questionnaire used in the case study interviews

Section 1 - Questions to acquire information about the practiced software process in the project:

?? What are the sub-processes that exist in the development process?

?? What are the inputs of the sub-processes?

?? What are the outputs of the sub-process?

?? What documents are produced in the end of the sub-process?

?? What are the documents that are needed to start activities in the sub-process?

?? How do you describe the formal/informal communication rules inside the process?

?? What are the reporting rules?

Section 2 – Questions to acquire information about employee’s awareness of the software process

?? What is your opinion about the processes and the process work?

?? Have you read the software development process description of the project?

Section 3 - Questions to acquire employee’s feedback on the practiced process

?? What are the problems in the software process that you are practicing?

?? What should be improved and how?

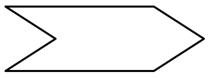
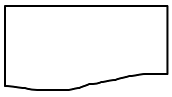
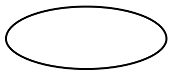
7.4. Process map

The process map presented in this section is a representation of the practiced software process at “ABC”. The process map was constructed based on the descriptions of the practiced software process provided by the organization’s employees during the interviews.

A process mapping is a technique to describe a process by means of using the workflow diagrams and supporting text. A process map usually has a defined notation to present different parts of the process, so-called sub-processes, information flow and interface between different sub-processes. For more information on process mapping technique, please see [Hunt96].

7.4.1. Notation used in the process map

We have used the following notation elements to build a process map:

Process map element	**Notation used**
Sub-process	
Result of the process (such as the documents or the source code)	
People executing the process	

7.4.2. A process map

In the process map, the organization’s practiced software process is referred to as “ABC development process”. This is done to highlight that given process map does not completely cover such sub-processes of the organization’s practiced software process, such as software testing and maintenance.

The picture below presents the sub-processes of the organization’s software process, which will be described in detail later on:

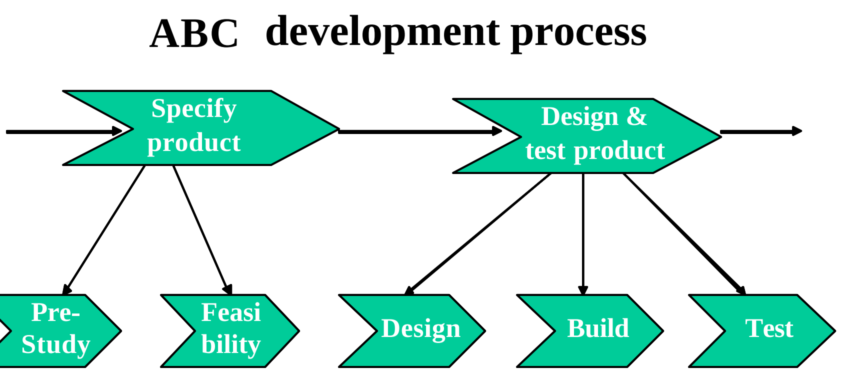


Fig.1

Pre- Study

Purpose: The purpose of this stage is to perform a technical analysis on the MRS and propose one or more alternatives of how to proceed with feasibility phase.

Besides of technical analysis, in the pre-study phase analysis on project cost and resources are also made.

Inputs: Main requirement specification (MRS), by SPM, acts as an input to the pre-study phase.

Outputs: As a result of that analysis, on the exit of pre-study several documents are created.

?? Project Specification (PS) by project manager

?? Project time plan by project manager (Not mentioned in process document)

?? Project milestone plan (MP) by project manager (Not mentioned in process document)

?? Technical investigation (TI) by SA or designer

?? Technical report (TR), which is a collection of the technical investigations.

Participants: SPM, System management, PM, TPM, SA, Designers, project CM, software CM.

Interfaces: PM or TPM acts as an interface between SPM and project members. In rare cases project members directly communicate with SPM

Sub-processes: Not identified

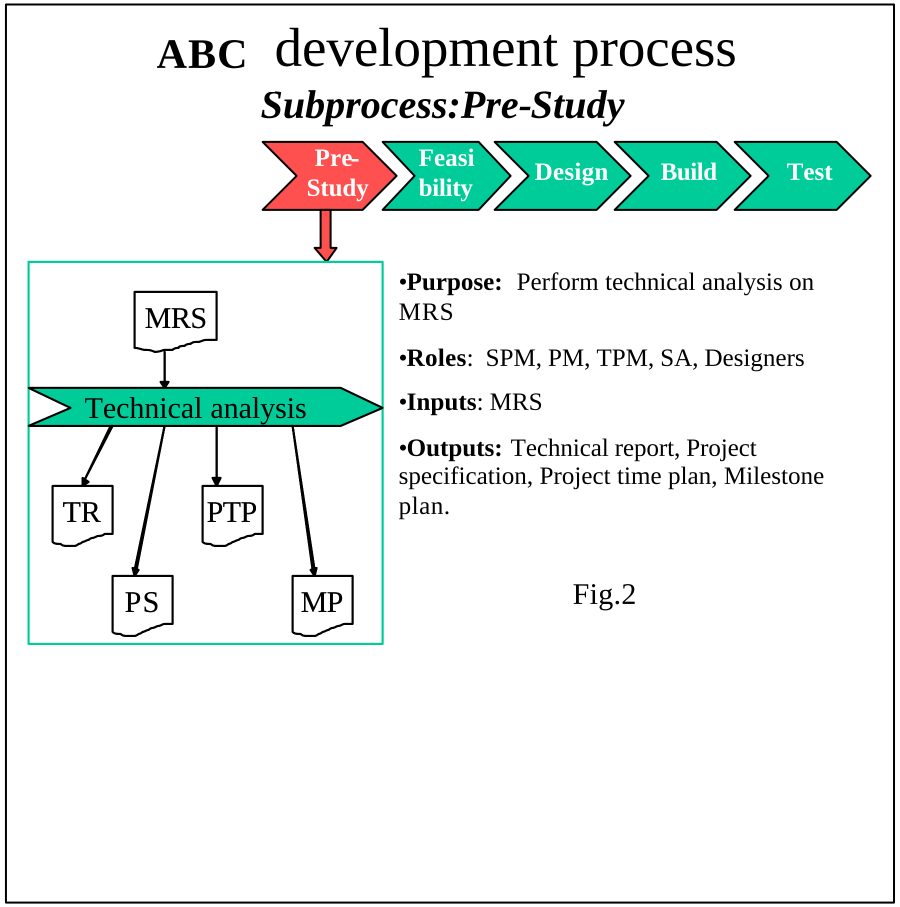

ABC development process
Subprocess:Pre-Study
Pre-Study
Feasi bility
Design
Build
Test
MRS
Technical analysis
TR
PTP
PS
MP
•Purpose: Perform technical analysis on MRS
•Roles: SPM, PM, TPM, SA, Designers
•Inputs: MRS
•Outputs: Technical report, Project specification, Project time plan, Milestone plan.

Fig.2

Inside Pre- study

Figure 3 represents the workflow in pre-study phase described by the employees.

In the schema on figure 3 everything starts with SPM, who writes MRS and gives it to PM and TPM. (Usually MRS is sent by mail).

PM/ TPM reviews MRS, and in case there are some uncertainties about the information in MPS they meet MRS to clear out the questions. After that, designers in the project group receive the assignments to pursue technical investigation on the requirements that are stated in MRS.

Before getting started with the technical investigation, a designer has to make it clear what requirement is about. To be able to do so he/she has to go around and ask questions to fellow designers and TPM or PM. In some cases the designer has to contact SPM in order to clear the goal of the requirement.

When the designer is finished with the technical investigation it is reviewed by a system architect. After that, the technical investigation is reviewed at TI approval meeting. All approved TI –s are gathered in TR (Technical report).

When TR is ready it goes back to SPM, to see that the project is on its track.

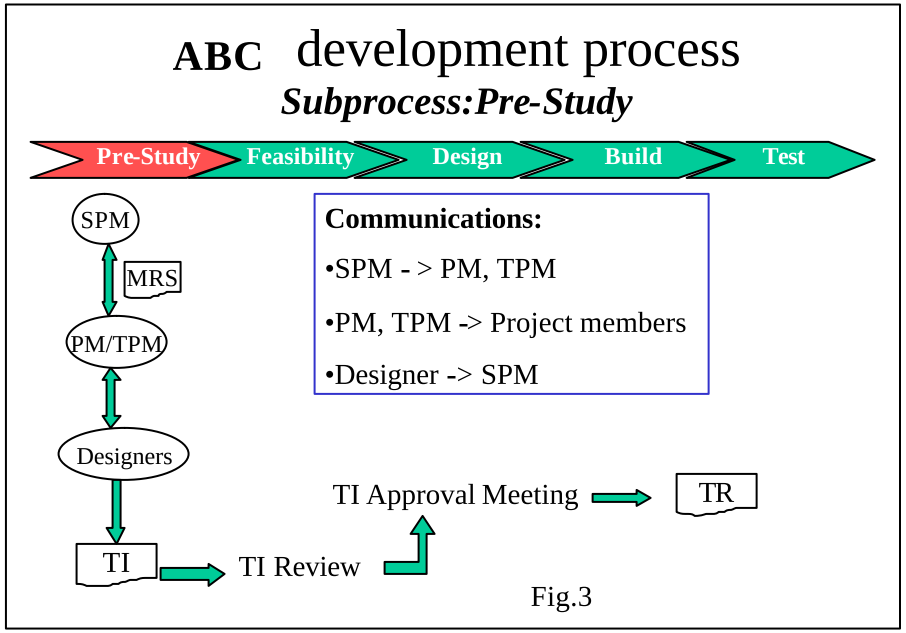


Fig.3

Feasibility

Purpose: The purpose of this stage is to produce implementation proposals, which will act as an input for the next stage.

Inputs:

- ?? Main requirement specification (MRS)
- ?? Technical Report (TR)

Outputs: This stage has only one output, namely Implementation proposal (IP).

Participants: PM, TPM, SA, Designers, Test manager

Interfaces: In feasibility there is direct communication between project members and TPM/PM

Sub-processes: Not identified

Notes: We have to note that the Test department starts to cooperate with the project in the feasibility stage. Testers are using IP-s to start designing test specifications for the software product.

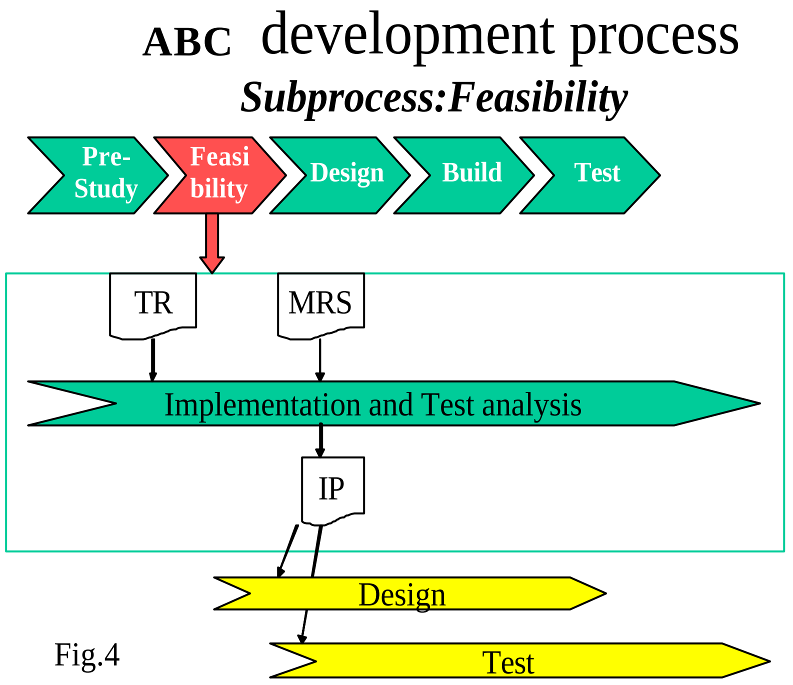


Fig.4

Inside Feasibility

Figure 5 represents the workflow in feasibility phase described by the employees.

Technical report acts as an input to the feasibility phase.

PM/ TPM gives an assignment to the designers to create an implementation proposal for the requirements that are pointed out in the technical report.

Before getting started with IP, a designer has to make it clear what is the goal of the requirement.

To be able to do so, he/she has to go around and ask questions to fellow designers and TPM or PM.

A very important information at this stage is to know what tools should be used in implementation.

Quotes from interview with the designer:

“One main task when you are getting started is to find out what you are really supposed to do. Once you have a clear picture what the goal is, then you have to check out another big question before you can really get started. This question is what tools are you supposed to use for this task”

When designer is finished with the IP, a system architect and a technical project manager inspect it. When IP successfully passes inspection it is ready to be an input to the next stage of development.

ABC development process

Subprocess:Feasibility

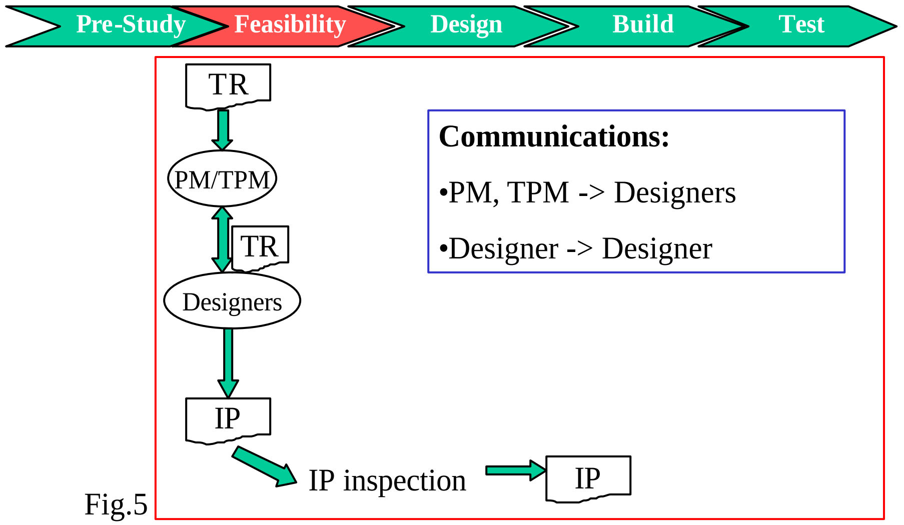


Fig.5

Design & test product

Purpose: To implement and test the product that is going to be handed in to the customer.

Inputs:

?? Main requirement specification (MRS)

?? Implementation proposal (IP)

Outputs:

?? Design Specification (DS)

?? Source code

?? Change impact record (CIR)

?? Modification description (MD)

Participants: PM, TPM, SA, Designers, software CM, test department

Interfaces: The software CM and TPM/PM often act as an interface between the designers and testers. This does not mean that the designers and testers don't communicate directly. At this stage of the development, the designers and the testers have quite close communication.

Sub-processes: Design and implementation; Build; Test.

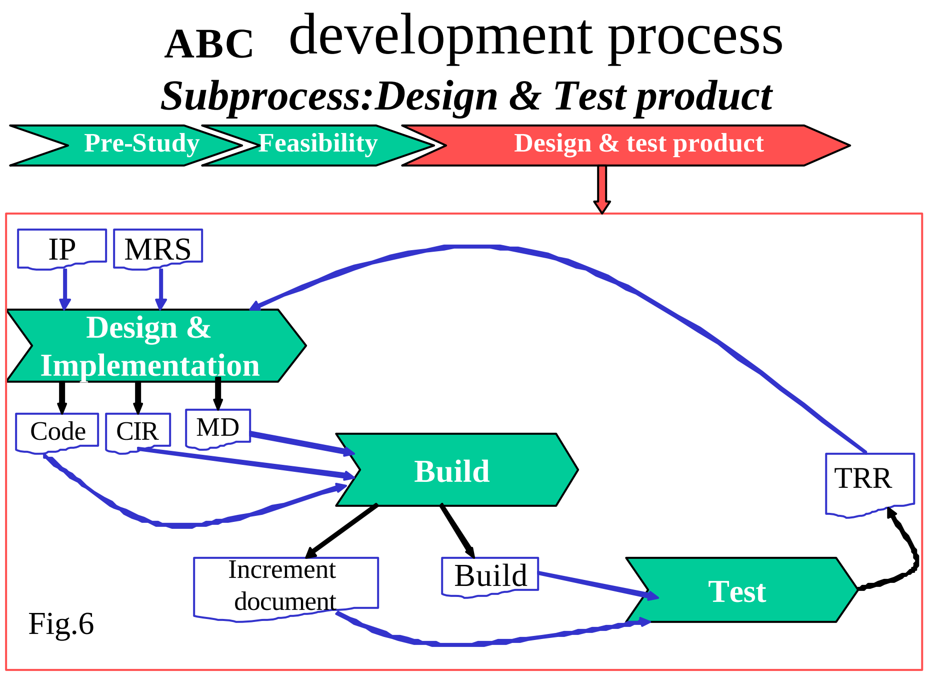


Fig.6

Inside Design and implementation

Figure 7 represents the workflow in "design and implementation" phase described by the employees.

In the schema on figure 7 everything starts with the implementation proposal, which acts as an input to this stage of development process.

Different designers are assigned to create the design specifications on the bases of IP-s. The designer starts with reading IP and making experimental coding to get the ideas about the design.

Then he /she create a design specification sketch, which is in most of the cases the copy of implementation proposal with some suggestions on direction for design.

After this, a DS sketch is communicated to SA, so that SA knows that designer has chosen a correct technical direction.

This process is called "Technical design review" in the current development process document for "ABC". Minutes of meeting are not documented.

As the design direction is approved, a designer starts the implementation phase, where he/she is creating the product's source code and also automated test modules in order to perform the basic tests.

During the implementation phase a designer has frequent communication with SA.

A designer uses automated test programs to test the code first on the local machine and when test results are satisfactory; source code (together with test programs) is checked in Daily build and compiled with all of the system.

Subprocess:Design & Implementation

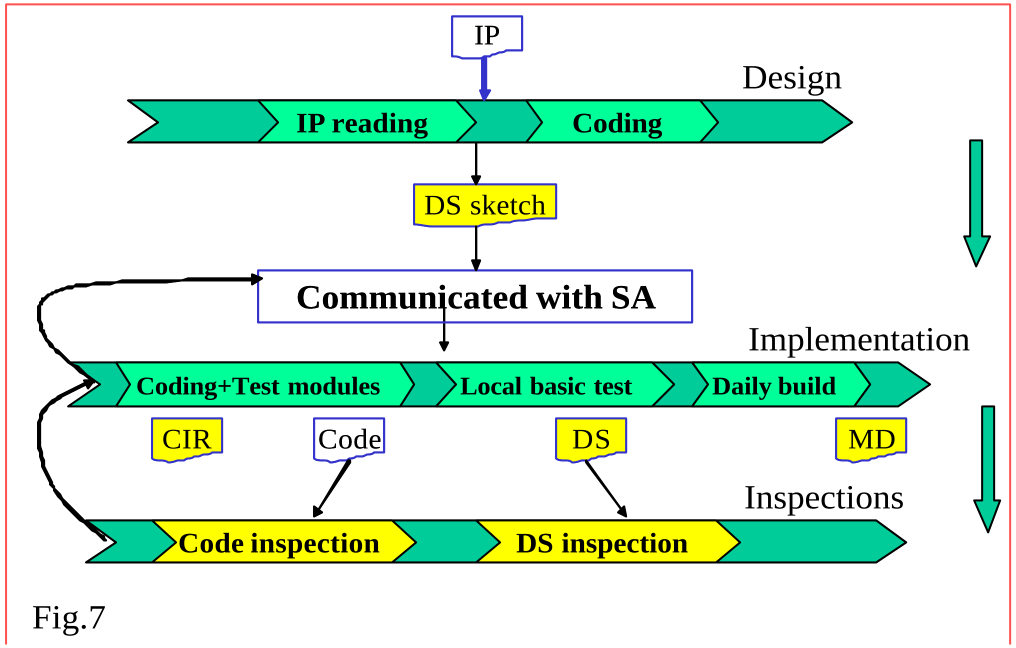


Fig.7

When the designer has updated the DS with the latest design details, before handling the source code to the test department, the source code and DS should be reviewed.

It is clear from the description of the process, that the source code and DS are outputs of this process. The source code is acting as an input to the next stage of the development, namely test, and DS is used in maintenance.

Besides source code and DS, other output is requited from design- implementation phase also. Those are documents called CIR (change impact record) and Modification description (MD).

Both of those documents are connected with the incremental nature of the development process, which is used in "ABC".

CIR and MD are used in "ABC"'s build creation process.

Inside Build

Activities in sub-process “Build” are carried out by the software CM. The purpose of those activities is to create the executable package, so called “build of ABC product” at the end of each increment.

To be able to do so, the software CM in cooperation with the project management identifies the date, when all the functionally that is planned for the given interment, should be checked-in. After that, the software CM freezes the code in the condition, as it was on that specific day and creates an executable build of “ABC” product.

To be able to succeed with this process the date when Freeze of the code is planned should be carefully communicated to the project members. The project members are informed about the freeze date on project meetings, and by mail also.

Another responsibility of the CM in this process is to track what the designers have created in every build. The information on each new functionality that has been implemented, or the changes that have been made on the existing functionalities has to be recorded. This information is crucial for the test department to carry out testing successfully.

The software CM tries to include the description of all the changes that have been made in the current build, in the “Increment document”, which is later used as an input to the test phase.

Before each freeze, the software CM sends a mail to the designers asking for the description of all changes that have been made in the current build. The designers have to put information about changes and modifications in the document “Modification description” and send it to software CM.

There is another document also that the designers have to write when they are correcting bugs, that were reported in trouble reports. This document is called change impact record (CIR). Information from CIR is also used as an input to the increment document.

Inside Test

We will not discuss the entire test process in this section. We will only concentrate on the interface between the development project and the test processes.

The first input to the test process from the project is at feasibility stage, when the test department can use the IP-s to create the test specification. Later on, when the build is ready, the test department gets the increment document and the build itself.

After testing the current build is complete, the test department returns to the project the trouble reports, containing information about found bugs in the build. Those bugs are thought to be corrected in the following build of the product.