Software Acquisition - a guide to areas of interest for the one-shot acquirer

Dan Ekström

Department of Communication Systems

Abstract

Many companies and authorities of today see great opportunities in using transaction-based software for information handling systems. During the 1990ies large companies and several authorities in Sweden have acquired such software. Unfortunately, a number of acquisitions have resulted in cost overruns and delays. The projects are similar, and in many cases it could be figured out, by experts, where and why the projects took a wrong turn. In the last ten years, the notorious capability maturity model, as well as other standards, has been introduced to support the acquirer. Why does projects still fail? Acquirers that make a one-shot acquisition for a small-scale information system does not have the knowledge to comply with such comprehensive guidelines. Standards are hard to apply since they appear to be immense for supporting acquisition of small-scale information systems. The standards however do comprise areas that could be modified to fit into the area of small-scale information systems.

The purpose of this thesis is to examine the area of software acquisition from the perspective of a one-shot software acquirer, acquring a small-scale information system.

The study first addressed a stakeholder of the thesis who recently finished an acquisition project. An open interview was carried out to set the context of the thesis. Literature known to support large acquisitions was analyzed with the intension of finding parts that may apply to the inexperienced one-shot acquirer. Then interviews with local authorities involved in acquisition projects were performed, giving insight into real-world problems. The analysis conclude in a recommendation for acquisition projects of small-scale information systems realized as a checklist.

CHAPTER 1

Introduction

Most companies and authorities of today see great opportunities in being represented on the Internet. During the 1990ies almost every major company and authority in Sweden have acquired software for Internet presence. Even companies and authorities that do not see the benefit of this feel an obligation to keep up with new offerings of technology. Unfortunately, many procurements have resulted in cost overruns and delays.

To stop acquiring software or continue in the same manner will not solve the problems. Common sense says that suppliers of software improve, or vanish from the market after making mistakes. The acquirer is often not interested in waiting for suppliers to improve and neither to acquire software under experimental conditions. Does it have to be this way? No, if the acquirer simply is aware of many of the problems, this is a starting point for making more accurate decisions.

Standards that recommend best practice exist in the area of software acquisition. They are hard to apply since they appear to be immense for supporting acquisition of small-scale information systems. Acquirers that make a one-shot acquisition for a small-scale information system does not have the knowledge to comply with standards. The standards do comprise areas that could be modified to fit into small-scale information systems. The knowledge that the standards comprise should be possible to be interpreted for a one-shot acquirer, if expressed with less abstraction.

1.1 Background of research area

1.1.1 Software

The term software does not only comprise computer programs. This view is to restrictive according to Sommerville [7]. From a wider perspective, software also comprises associated documents and configuration data, which are needed to make the software operate properly. Software often consists of several programs, configuration files, system documentation and user documentation. Sometimes tutoring for end-users is included in the price of the software.

According to Sommerville [7] software products can be of two types:

- *Generic products* These are stand-alone systems which are produced by a development organization and sold on the open market to any customer who is able to buy them. Sometimes they are referred to as shrink-wrapped software. Examples are database tools, word processors, drawing packages and project management tools.
- *Customised products* These systems are commissioned by a particular customer. It is specially developed for that customer by a software contractor. Examples of this type of software include control systems for electronic devices, systems written to support a particular business process and air traffic control systems.

An important difference between these different types of software is that in the generic products the requirements specification is controlled by the company, which develops the software. In custom projects the specification is usually controlled by the organization that is buying the software. The engineering discipline used for development of software is briefly introduced below.

1.1.2 Software Engineering

Software Engineering is defined in Sommerville [7] as an engineering discipline, which is concerned with all aspects of software development. From the early stages of the system specification through to maintaining the system after it has gone into use. In general, software engineers adopt a systematic and organized approach to their work as this is often the most effective way to produce high-quality software.

The software should have certain attribute to meet the criteria of high-quality software according to Sommerville in [7]:

- *Maintainability* Software should be written in such a way that it may evolve to meet the changing needs of customers. This is a critical attribute because the original need for the software changes as an inevitable consequence of a changing business environment.
- *Dependability* Software dependability has a range of characteristics, including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of a system failure.
- *Efficiency* Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilization, etc.
- *Usability* Software must be usable, without undue effort, by the type of user for whom it is designed. This means that it should have an appropriate user interface and adequate documentation.

1.1.3 Software Acquisition

Acquisition of software is a complex discipline that may require knowledge of several other disciplines.

- *Requirements engineering* The acquirer may want to elicit requirements optimized for selection of the most appropriate supplier with the most appropriate product. Requirements engineering could help to influence the product quality by specifying quality requirements and work process requirements. It is used for specification of features, business environment and end-user everyday tasks.
- *Communication interfaces* The discipline of interplay between a supplier and an acquirer. The interface between the acquirer and supplier is present on several levels of the acquisition. More or less communication is more or less expensive for the acquirer, but may benefit to improve the users acceptance of the product.
- *Verification and validation* The acquirer must be able to verify and validate the product. Validation is to see if the product contains the specified features. Verification of the product is to see that the features works as defined in the specification.

1.2 Problem

1.2.1 Background

The examples below indicate that procurements often result in cost overruns, delays or that the final product has poor correspondence with requirements.

A similar investigation reveals indicia for the same results in the United States. The Standish Group [27] has investigated large, medium, and small companies across major industry segments, e.g., banking, securities, manufacturing, retail, wholesale, heath care, insurance, services, and local, state, and federal organizations. The total sample size was 365 respondents and represented 8,380 applications. The survey reveals that on the success side, the average is only 16.2% for software projects that are completed on time and on budget. In the larger companies, the news is even worse: only 9% of their projects come in on time and on budget. Even when these projects are completed, many of them has not fulfilled specification requirements. Projects completed by the largest American companies have only approximately 42% of the originally-proposed features and functions. Smaller companies do much better. A total of 78.4% of their software projects will get deployed with at least 74.2% of their original features and functions.

In 1998 the Swedish government carried out an inquiry about IT-development in governmental departments [5]. The inquiry scoped 231 projects. More than half of the 231 projects had been subject to serious problems with time plan or budget. The investigation made it plain that if a project had problems with budget or time plan, it was likely that the project would be affected by the same problems again. It also revealed that problems are more extensive in projects where subcontractors are used and in projects with large budgets.

The authority of Lund, a city in Sweden, have recently launched a new web site. The request for bids on the job was launched during Spring 1999. 7 different alternatives, ranging from 264 000 SEK to 2 750 000 SEK, were offered from different companies. The second cheapest was chosen, and recently, 4-5 months delayed, the website was finally launched.

The Swedish national authority of superannuation, PPM, prepared and completed an acquisition of an administrative information system [4]. The information system was built on a web based solution with the user interface encapsulated in a web browser. After a year of development PPM requested a change of Swedish legislature in order to make it possible for the supplier to complete its commitment. At this point PPM had lost its faith in the supplier and started an inhouse development

project to expand the already existing information system with similar functionality. Development continued in parallel until the inhouse project was released for public use.

The problem domain has a worldwide range. Investigations in the United States reveals similar figures as investigations i Sweden. The two last cases are only two of several not so successful projects carried out in Sweden in the 90:ies.

1.2.2 Issues for investigation

The success of acquiring a small-scale information system is nothing to take for granted. In fact the investigation about IT-development in governmental departments in [5] shows that projects with a small budget is less likely to succeed.

In [5] small projects are unique in the sense that they were all underestimated in degree of difficulty and the resources for guaranteeing a quality product were insufficient. In many projects resources allocated to planning were inadequate and hence the projects planning were inferior. Many of the problems that the projects experienced could have been solved in planning before start of the projects.

According to [5] the outcome in these cases are likely to depend on: the acquirers are inexperienced, the qualitative resources allocated are insufficient due to underestimation of difficulty to acquire software and little understanding of the resources that the authority should dispose or which resources that the authority should subcontract.

The conclusions in [5] where the same conclusions as in "Bättre ADB-projekt om hur man undviker de vanligaste fällorna vid utveckling av ADB-system" (RRV 1994:31):

1. A flexible organization for the project should be created.
2. Areas of responsibilities should be defined.
3. Planning should be conducted with great accuracy.
4. A method of development should be chosen and followed throughout the project.
5. The process of development should be evaluated for its ability to contribute to quality.
6. Evolution of costs should be followed and controlled.

7. The integration of the system in the organization should be planned in an early stage.

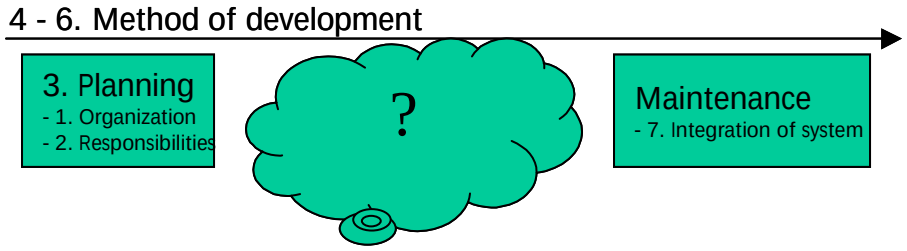

FIGURE 1. Illustration of the seven conclusions above

Figure 1 illustrates the conclusions above. Paragraph one and two are tasks that should be decided on in the planning phase. Paragraph three is the planning phase containing paragraph one and two. Paragraph four and five concern the method which the software projects uses and its proficiency to contribute to quality. Involved in the project method is the follow-up of costs, this is paragraph six. The last paragraph is the first step in the maintenance phase where the system is put into use.

The investigation carried out by the Standish group reveals three major success factors for projects that have been successful: user involvement, executive management support and clear statement of requirements.

In the paragraphs 1-7 phases that may be relevant to investigate are not mentioned. However they give a good starting point for issues to be stressed. The issues below should cover the proposed areas of improvement from [5] but also add issues that cover the areas that exists between Planning and Maintenance in Figure 1.

1. *Product or service - user involvement* In the Lund case section 1.2.1 costs in the winning bid where specified as if the web solution was a product that was going to be delivered at a certain date, without no customer involvement. Nevertheless, customer involvement is necessary since the product has to be tweaked to fit the context of each unique customer. Which tasks could the customer expect to perform? A customer representative could mean many things. What kind of personnel is relevant for which tasks? What are tacit costs of customer involvement?

2. *Communication* In the PPM case and the Lund case Section 1.2.1 a lack of communication seems to have caused many of the problems in the projects. Is it possible to make the customer fully aware in non-technical terms, with formal communication, exactly what progress is made while designing the product? Is some form of communication more useful than other when crossing the chasm between developers and customers? This may depend of the method used to develop the product?
3. *Existing support for acquisitions* Standards for acquisition exists. What are the strengths and weaknesses with such standards [3, 9].
4. *Requirements* In Section 1.2.1 in the Lund case, evaluations of the system and communication of functionality had no basis in documented requirements. Litterature reveals that this is essential for evaluating software [16] and having a common platform for communication [13]. This may have lead to co-operation problems and confusion for both supplier and acquirer. Is it enough to define requirements as use-cases or scenarios for evaluation of project progress and functionality of the acquired product?
5. *Non-existing support for acquisitions - a pragmatic approach for one-shot acquirers* For the one-shot acquirer a pragmatic guidance to acquisitions of software may help to gain understanding of why it is hard to acquire software and what could be done to reduce risk and concentrate effort on the right tasks. To produce such a guidance would require to examine the existing support for acquisitions and to distill the areas applicable to small-scale information systems. Further the areas not covered in the paragraphs 1 - 7 above will need an investigation. This may be performed by case studies and interviews. This pragmatic guidance is tightly associated with the objectives of the acquisition. Is the object to get a product that has covered all defined functionality, is delivered within budgeted cost or within schedule? Often the acquirer would like to optimize for one of the three options. Using the software acquisition guidance with different options may perform this optimization.

1.3 Purpose

Knowledge creates awareness and understanding. The buyer of software should be able to see the consequences of its actions, hence being able to plan and take control as far as possible.

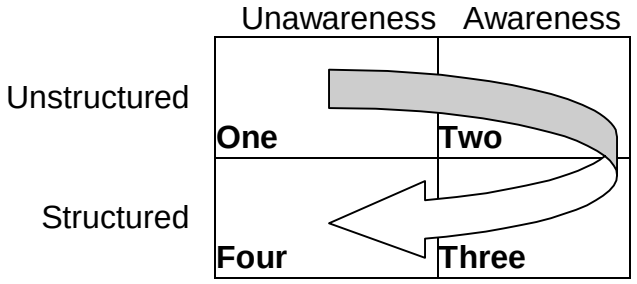

FIGURE 2. Model of learning to improve one self

Figure 2 displays four states. The states are similar to the process of learning to improve one self. In a state you can be aware/unaware and structured/unstructured. In state one you have not reflected over present circumstances as problems and you are not aware of any solutions either. In state two you define some of the circumstances as problems, but you are not aware of any solutions or countermeasures. State three is representative for being an experimental stage where new routines to handle problems are applied and evaluated. In the last state the knowledge and the countermeasures are established. The established routines then become routine tasks sliding into state one as continuum. All knowledge is not gained at once in once state and therefore the process consist of parallel iterations.

In the problem description Section 1.2 we learn about cases, that to some degree, reflect the current practice of acquisition in Sweden. On the analogy with the model for learning and improvement, current status is somewhere between state one and two, with awareness of the problems but without knowledge how to take countermeasures. It is important to realize where we stand today, by using a structured method to grasp the status of today in software acquisition. This makes it possible to bring out areas that would benefit from improvement.

This thesis intends to make a survey of current status and to identify areas that would benefit from improvement. This reflects state two in the model. The survey should conclude in suggestions to countermeasures, which also should be evaluated by application on a case study. This is area is between state two and state three. Hopefully readers will gain knowledge to experiment with, reflecting state three.

1.4 Scope

An acquisition is a cooperation between an acquirer and a supplier. The scope of this thesis is limited to the acquirer with its interface towards the supplier. The acquirer is characterized by:

- *One-shot software acquisition* Software acquisitions are not conducted on a regular basis. Software may only be acquired once every five years. No routines exist for buying software.
- *Experienced acquirer* Other things than software is acquired at regular bases, hence the acquirer is accustomed with acquisitions.

The procurement is limited to small-scale transaction based software systems. The system is a custom made system where the requirements are controlled by the organization that is buying the software. The software has the following characteristics:

- Transaction-based
- Moderate numeric processing requirements
- Small database
- Relatively flexible time constraints
- Flexible, complex user interfaces
- Requirements and design driven by user interface - must match way of doing business

These paragraphs characterize a small-scale information systems according to [1].

1.5 Target audience

The problem description and scope of the thesis is limited to the aspects of an experienced acquirer, which is not experienced with buying software. However the thesis is an academic work with the purpose to be included in my exam. The discussion in the report is a mixture for the two purposes.

1.6 Outline of the report

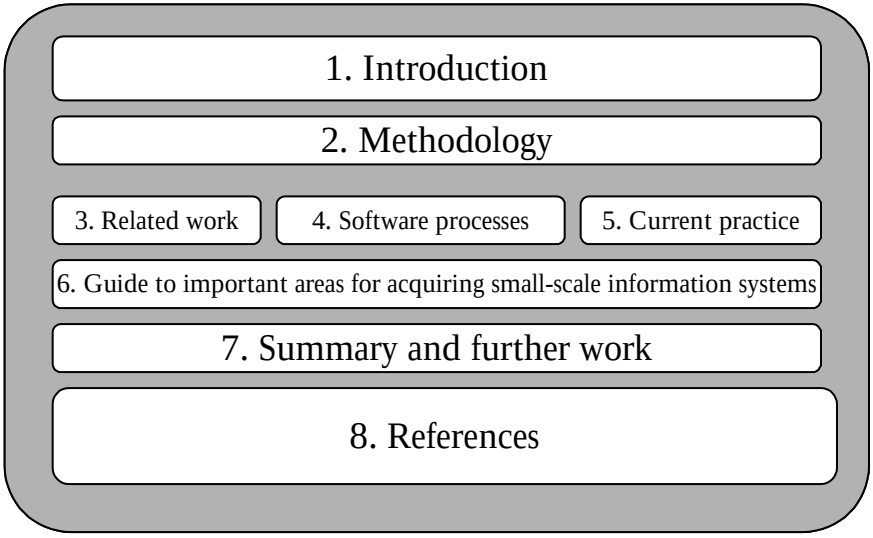

FIGURE 3. Outline of report

- *Chapter 1* is an introduction to the context of software acquisition. This includes background, problem description, purpose and limitations.
- *Chapter 2* describes methodology used in the thesis.

- *Chapter 3* identifies related work in the area. Standards for software acquisition are described.
- *Chapter 4* elaborates on software process since they affect the communication interface, acquirer - supplier, during implementation. They also deal differently with requirements implementation.
- *Chapter 5* contains conclusion and recommendations from the interviews performed.
- *Chapter 6* contains a guide to areas of interest, built on the recommendations from Chapter 3, 4 and 5.
- *Chapter 7* Summary of the thesis and pointers to further work.
- *Chapter 8* Contains references.
- *Appendix A* Listing of interviews.

CHAPTER 2

Methodology

In this chapter scientific methods and techniques will be described. There is a difference between methods and techniques according to Brandt in [23]. The method of working is about how the material is prepared. The scientific technique refers to how the material is collected.

The simplest of all methods is the case were the phenomenon is described in a running text. It is important that this method involves some kind of systematics. The facts that are collected must be categorized and sorted in some specific order. The material that is chosen to be described must fit into the context with the other text. It is important to constantly make selections from the collected material so that nothing irrelevant to the context is described. What is considered to be relevant depends on the purpose of the text. The description must not only be relevant but the contents must be genuine.

2.1 Techniques

2.1.1 Literature research

Literature comprehends: books, articles, reports, essays etc. Literature searching precedes literature studies. The studies are normally conducted at the library through a computer or librarian. If relevant literature is found, checking the references may be a fast way of finding further relevant literature.

2.1.2 Interviews

The form an interview may take ranges on a scale from open to structured. If the interview is entirely open, it means that the questions are formed so that the interviewee's thoughts not are bound to a specific type of answer. The interviewee may freely talk around the subject.

The interview may be entirely structured. The interviewer is already prepared with questions and the respondent chose from alternative answers. The most structured interview is the inquiry. A summary of the two forms as in [24]:

- *The open interview* is used to reflect the information subjective. It is used for understanding the information from the perspective of the interviewee. Focus is on how the individual thinks, on what meaning the individual gives the phenomenon. The interviewer chooses the subject but the interviewee lets the interviewer understand which areas in the subject that are important. The interviewee defines and limits the phenomenon. Hence different interviewee's may give different definitions of the phenomenon. This interview form makes it difficult to draw conclusions about quantities.
- *The structured interview* makes it possible to obtain information about quantities. In this form of interview the interviewer characterizes the phenomenon and decided on its limits. The interviewer decides the context. The interviewer gets the interviewee's perspective on the phenomenon but is not interested in how the interviewee relates it self to the phenomenon. The individual is not important in the situation. The interviewer is looking for relations among concepts rather than different meanings.

The questions do not decide the choice of interview form according to Merton Robert in [26]. The design of the possibilities for answers chooses the interview form.

2.2 Methods

2.2.1 Case studies

The purpose of a case study is to examine a small part of a lapse and let the case represent phenomenon in some aspects. The advantage is that this gives the reader a relatively clear picture of current status without having to describe the big scene. The disadvantage is to know weather the case studies is representative for the phenomenon or not, summarized in Brant Kjeld [23]. Hence conclusions drawn from the study may be indicia. The case studies may help the writer to grasp the subject. The writer may then gain experience in the topic by the case studies and they may help the writer to identify important issues. Examples of case studies are the study carried out by the Swedish government about acquisitions performed by local authorities [5] and the study made by the Standish Group in governmental acquisitions in the U.S. [27].

2.2.2 Classification

The classification must fill some logical requirements to be able to be used to analyze data and to be able to draw conclusions. The following requirements exists on the classification of the data:

1. The classes must be reliable.
2. They should be genuine.
3. The classes could easily be applied on the material without a to big part of the material ending up in a leftover class.
4. The classes should be mutual exclusive. The data should not fit into more than one class.

5. Emty classes should not exist. If empty classes exist, this is a sign of that the original classification not are very useful.

2.2.3 Quantifying of data

If data is quantifiable it may be treated statistical. With software statistical analysis may be applied. The quantitative data has a pedagogical advantage. The advantage is that it is easy to present to the reader. It may be displayed in tables and diagrams. It is easier to make comparisons with quantitative data analysis.

2.2.4 Comparison

This method compares phenomena with each other. The method is hard to apply since comparisons not may be performed straight off. The comparison may become invalid if some other variables also are included. Comparisons should be used carefully. Below are some considerations that should be made before using comparison as a method:

- Use phenomena that are comparable.
- The phenomena should be generalized to fit the context and to be able to compare.
- Similarities as well as dissimilarities should be described to display the whole picture.

2.2.5 Hypothesis and evaluation

The hypothesis is an assumption. The researcher makes a qualified assumption about certain circumstances. The assumption should be built on known facts to be of value. The hypotheisis is normally used for explaining a phenomenon that the researcher has studied. The researcher may try to prove the hypothesis to be correct or to eliminating hypothesis by falsifying them.

2.2.6 Theories

When forming a theory the researcher takes a starting point in known facts but wants to explain some phenomenon that not could be understood. From the known facts hypothesis are derived. With these hypothesis and other known hypothesis new hypothesis are formed. They are held together with a net of derived hypothesis and facts.

2.2.7 Model

When forming a model theories and hypothesis are used. But they must be used to mirror reality. A complicated model is more likely to mirror reality. The level of complexity of the model should mirror the utilization of the model.

2.3 Practical methodology

Initially literature studies of the context of acquisitions of software will take place. These studies will lead to grasping the problem domain refining the problem description and forming the scope of the thesis. The thesis will then contain deeper understanding of the area summarizing the information given from the deeper literature studies.

By the deeper knowledge coming from the literature studies the ground is prepared for interviews. The form of interview that will serve the purpose of examining current practice of software acquisition is described below.

The interviewer should mainly decided the context. This could be done formally before the interview or by identifying new parts of the evolving context during the interview. The organization of the interview should be restrained to questions in certain areas, hence not following the interviewee's thoughts. This approach is taken since by the literature studies the frame of interest is set for the thesis. The interviewee may give hints of new areas of interest during the interview but they should then lie in the frame of interest defined by the literature studies. Hence the interviewee then gives answers to what the interviewer finds to be meaningful to the context.

To make the organization of the interviews a bit more open the interviewer could lead the interviewee into the right area and then work with follow-up ques-

tions. The analysis afterwards could show that several of the interviews were dissimilar which shows the variety of qualities that the area possesses. If interviews with similar theme turn out to be comparable it could be analyzed quantitatively. But the open type of answers could limit the possibilities of a qualitative analysis of the qualities of the phenomenon.

The interviews's pupose are to be able to see the area with the eyes of a real-work acquirer. This hopefully will reveal real-world problems. The case studies will be used for pointing out unique problems in the eyes of the acquirer. If common problems are found they will be treated in the chapter for conclusions. Finally the chapter with areas of interest for the small-scale software acquierer will contain a compilation of the indicia from standards and interviews to areas that needs improvemen and why.

CHAPTER 3

Related work

3.1 Standards for software acquisition

Standards are important of several reasons according to Sommerville in [7]:

- They provide an encapsulation of most appropriate practice. The knowledge is often only acquired after a great deal of trial and error. Building it into a standard avoids the repetition of past mistakes.
- They provide a framework around which the organization may implement it's own best practice.
- Standards assist in continuity where work carried out by one person is taken up and continued by another.

Several standards exist in the area of software acquisition. Two prominent standards are examined. The "Software Acquisition Capability Maturity Model (SA-CMM)" [9] is developed by the Software Engineering Institute (SEI). It is known to be used by large corporations with stringent requirements on quality of software. Some of the frequent users of SA-CMM are known to be sponsors of SEI.

The other standard is [3] "Recommended Practice for Software Acquisition (RPSA)". It is developed by the Institute of Electrical and Electronics Engineers

(IEEE). IEEE develops standards for public use to promote electrical technologies and sciences.

The two standards has several common areas. The table below (Table 1) displays key areas from the two standards. All key areas in the standards comprise almost the same features. The comparison is between SA-CMM level 2, which is the repeatable level, together with RPSA.

The purpose of SA-CMM is to establish and improve the process which is used to acquire software. The purpose of RPSA is to provide a general framework which should be interpreted with basis in the own organization. The standards are ment to address large acquisition projects in large organizations.

SA-CMM (Level 2)	Common denominator	IEEE RPSA
Software Acquisition Planning	Early budgetary action, schedule determination, acquisition strategy and risk indentification	1) Planning organizational strategy
Solicitation	Select a contractor who is best capable of satisfying requirements of the contract	4) Identifying potential suppliers 6) Evaluating proposals and selecting the supplier
Requirements development and management	Develop and maintain requirements	3) Determining the software requirements 5) Preparing contract requirements
Project management	Manage schedule, cost and effiency	
Contract tracking and oversight	Ensures that the software activities under contract are being performed in accordance with contractual requirements	7) Managing suppliers performance
Evaluation	To dermine that the aquired software product and services satisfy contract requirements	8) Accepting the software
Transition to support	To provide for the transition of the software products being acquired to the eventual software support organization	9) Using the software

TABLE 1. Common denominators for SA-CMM and RPSA

Table 1 shows the common denominators of SA-CMM and RPSA. These areas are generically important for software acquisition as they comprise the entire process step by step.

Below the reader is first introduced to each of the two standards. Then follows after each of the standards, a section that emphasises what in the standards that could be used for further work in adapting them to small-scale information systems.

3.2 Software Acquisition Capability Maturity Model

3.2.1 Description

The Software Engineering Institute (SEI) of Carnegie Mellon University developed the CMM of software process improvement during the late eighties and nineties [6]. The major sponsor has been the US Department of Defense (DoD). The main reason for this was the need to get control over cost and quality of the software products delivered by its contractors. The intellectual framework was first presented in the work "Capability Maturity Model for Software, Version 1.1" [17].

The key words of CMM are maturity and capability. The basic assumption permeating the Capability Maturity Model is that software development organizations with a structured way of doing their job will over time deliver software with higher quality than less well structured organizations will. In CMM terminology this means that a well-structured organization has better capabilities to deliver high quality software. CMM offers a concept of:

- How an organization can get well structured.
- Where to start and how to control the organization evolution process?

In CMM terminology this would be expressed: how an organization gains maturity. Maturity and capabilities are very central to the model. Another very central concept is the Key Process Areas. The Key Process Areas are the building blocks

of each capability level. The levels of maturity are initial, repeatable, defined, managed and optimizing.

- *The initial level* Represents an organization that has no stable environment and great difficulties to make commitments according to when and how they are doing what. If there was a plan it is abandoned when crises arise. Sometimes the organization succeeds in its project but it does not have the capability to learn from neither its successes nor shortcomings.
- *The repeatable level* The organization has learned to make plans based on experience of previous similar projects making it easier to make realistic project commitments. The project management tracks project cost, schedule and product functionality. The concept of controlled baselines is introduced. The relations to customers and subcontractors are not so chaotic. The development process is repeatable and the capability is said to be disciplined.
- *Defined level* The standard process for development and management in the organization are documented. A software acquisition process group is established and is responsible for the process activities. The project process is an instance of the standard process and includes factors like readiness criteria, document standards, verification mechanisms and activities completion criteria. At this defined process level the project management has insight in process and acquisition status on an almost real time basis. The capability of the acquisition organization is now said to be standard and consistent.
- *Managed level* Level four is characterized by quantitative measures. The organization is able to sets quantitative quality goals and measure productivity and process to compare between and across projects. The measures are stored in an organization-wide databank. The projects are in full control of their acquisitions. The quantitative variations fall within acceptable limits. The ability to handle surprises since qualified risk management is established. The capability at this managed level is quantifiable and predictable.
- On the *optimizing* level the development organization is established in a dynamic continuous evolution mode. Process improvement is the natural state. Thanks to the metrics assessed since level four the organization has gained the ability to make cost-benefit analysis as decisional ground for process improvements. The capability now is planned and controlled.

Guidance to improvement in achieving higher maturity in acquiring software is a characteristic of the SA-CMM. These levels of maturity are found in the SA-CMM.

The SA-CMM [9] has several areas in interest for the acquirer. The SA-CMM contains the following key process areas on level two, the repeatable level:

- *Solicitation* The purpose of solicitation is to perform the activities necessary to prepare for the evaluation of responses, conducting the evaluation, conducting negotiations and awarding the contract.
- *Requirements development and management* The purpose is to establish a common and unambiguous definition of software-related contractual requirements that is understood by all parties involved. This key area of level two are divided into two parts where one is the development of requirements and the other is management of the requirements for the duration of the acquisition. The development involves decomposing system level requirements into software requirements. Software requirements management involves establishing and maintaining agreement among the participants. Baselining software requirements and control subsequent requirement changes.
- *Project management* This involves planning, organizing, staffing, directing, and controlling project office activities. The purpose of this is to endure that the software activities under the contract are being performed in accordance with the contractual requirements and that evolving software will satisfy contractual requirements.
- *Transition for support* The purpose of this activity is to provide for the transition to the eventual software support organization.
- *Acquisition management* Risks should be identified as early as possible and the acquisition strategy should be adjusted to manage those risks. Acquisition risk management is a two-part process. First the software acquisition strategy identifies the risks associated with the acquisition and the approach is planned based on those risks.
- *Training program* The training program should develop skills and knowledge of individuals so they can perform their software acquisition roles effectively and efficiently.

Further reading in this area has recently emerged, as the CMM [6] and SA-CMM [9] is integrated in [11], which is an early draft version of the CMM-Inte-

grated. This is an effort to merge the different branches of the CMM theories framework.

3.3 Application of SA-CMM to small-scale information system

The key process areas of the SA-CMM consist of goals, commitments, abilities and activities.

The description in Section 3.2.1 is concerned with the key process areas on level two. At this level, "The repeatable level", the project team is aware of supportive of policies, regulations and standards that relates to the project and makes a dedicated attempt to comply with them. The teams should be able to transfer successful practices from earlier project to new ones. This feature and several other features of level two may be useful for the small-scale information system acquirer. The following sections contain characteristics from each key process area that should be of value for a one-shot software acquirer of a small-scale information system.

3.3.1 Planning

One of the goals with this key process area is that the plans should encompass the total software acquisition effort. This is important for several reasons. The cases in Section 1.2.1 the acquirers had much confidence in the competence of the suppliers. This could lead to less desire for the acquirer to make plans for the project, since it may be easier to give the responsibility to the supplier. If this is the case, true needs and understanding of the own interest in the acquisition may be blurred in the contact with the supplier.

The other goal is that the documents of planning are prepared early in the acquisition process prior to actions that involves contractual software acquisition efforts. One of the activities of this phase is that a strategy should be developed and documented. The strategy should include:

- Objectives of the acquisition
- Project constraints, such as funding and schedules.
- Available and projected technologies

- Software acquisition methods
- Potential contract types and terms
- End-user considerations
- Risk identification

Among the paragraphs above, risks identification is present in the strategy-phase, opposed to other standards. Awareness of possible outcomes of the projects, risks or not, is healthy and may contribute to planned countermeasures or unique maneuvers. It is also important to commence a training program for the persons involved in the project, for the reason that they should be able to identify anomalies. Another important activity is that the planning is documented and maintained over the life of the project. If documentation is poorly maintained to mirror the changes in the project, then the project members most likely will not turn to the documentation for advice. This could be important since it lies in all parties' interest that project members for example keep the requirements specification current in mind. The planning should involve these paragraphs:

- The software parts of the project
  - Risk identification
  - Management
  - Solicitation
  - Requirements development management
  - Evaluation and transition support
- The tasks to be performed
- The required resources including funding, staff, equipment and tools.
- Roles for project members
- Master schedule of acquisition milestones
- Measurement to determine the progress of the acquisition

Costs of lifecycle support should be included in the software acquisition planning documentation. This is important because maintaining software is very expensive. It may cost as much as to four times developing a new system according to Sommerville in [7].

Costs estimated for the software products and services being acquired should be prepared and independently reviewed. It is important to review the costs of the services part of the product since they may not be as obvious as the product price.

3.3.2 Solicitation

In the phase of solicitation genuine requirements are prepared and reviewed to be of high quality.

An important commitment of solicitation is that a person exists that is responsible for the selection process and the selection decision. The goal of this key process area is that a contractor who is qualified to satisfy the contract's requirements for the project's software-related products and services is selected.

Several activities of importance for the small-scale information system acquirer exist in the solicitation phase. A solicitation plan should be prepared. The plan should contain the requirements of the software. Except for the requirements, other software-related tasks should be documented. For example: evaluation tasks and documentation tasks. Costs and schedule estimates from planning should be further refined.

The project team should take action to ensure that all parts involved have a mutual understanding of the project requirements. If the acquirer totaly agrees on requirements internally before contracting the supplier, many risks of misunderstandings are eliminated. The focus may then be set on introducing the supplier to the domain. The requirements should be reviewed to reflect the real needs of the organization. It is expensive to find out that the software specified not fills the need of the users.

3.3.3 Requirements development and management

The goals of this key process area are:

- To maintain and develop requirements in conjunction with stakeholders.
- Baselines for requirements should be established and managed.
- Keep contractual requirements unambiguous

It may be important to create a common platform to work from if sub-projects use the same requirements as basis. The list below describes important activities for this key process area:

- To identify groups and intergroup communication associated with requirements development.
- Define procedures on how to work with requirements development.
- Define procedures for requirements management and baseline establishment.
- Documents procedures for how to handle impact analysis, when requirements affect each other.
- Keep a documented tractability between the status of software and the requirements.

These paragraphs make sure that requirements are not lost track of during the project. It is essential to know which requirements are implemented as working features and that the supplier receives uniform feedback on requirements even if several end-users are verifying the same requirement. This mapping of requirements communication paths is also good for keeping track of the requirements that is elicited during the project. It may be hard to assess effects of the new requirements being added adhoc during the project. If kept track of they may be added in the next iteration or the next project.

3.3.4 Project management

The goals are to manage problems, costs, schedule and objectives. The project team should perform its activities according to a management plan. The plan could contains these paragraphs:

- Project objectives, purpose, scope and duration
- Project team structure, roles and responsibilities
- Acquisition strategy
- Project performance, cost and schedule baselines
- Coordination and communication
- Risk identification and tracking
- Software engineering approach

- Software support requirements
- Security policy and requirements
- Corrective action reporting procedures
- The extent of the end user involvement in the acquisition

The above paragraphs are similar to the ones defined in Section 3.3.1, planning. The project should be managed to fulfill the project plan. The project management plan is an action plan for use on the project planning. The two plans could probably be combined if the project is small and no confusion could arise.

3.3.5 Contract tracking and oversight

The main objective with contract tracking and oversight is to see that the software product and services satisfy contractual requirements. The software engineering effort should comply with the contract requirements. A plan for the execution of the tracking should exist. It should contain identification of groups, assigned responsibilities, intergroup coordination and the extent of end user involvement. The project team should also review the suppliers' software planning documents. Changes by the supplier in the suppliers' software planning documents should be coordinated in with the acquirer.

3.3.6 Evaluation

The objective of this key process area is to be able to provide an objective basis to support decisions for accepting the software products and services. The acquirer should have a written policy for managing the evaluation of the acquired software products and services. The projects evaluation requirements should develop during the lifetime of the project. This goal may be hard achieve at a late stage in the project as many of the features has become matters of course. Since new ideas have arised during the project it might be hard to accept the product in its standard edition. The planned evaluations should however be performed prior the acceptance for operational use.

3.3.7 Transition to support

The goal is for the software support organization to have the adequate capacity to be able to give support. For example there should not be any loss in continuity of the support if it is transferred between organizations.

3.3.8 Summary

The SA-CMM has a comprehensive view of the whole process of software acquisition from creating strategies to transition to support. However it would need modification to fit into the context described in Chapter 1. The sections above emphasize parts of the SA-CMM that would apply well even to acquisition of small-scale information system. Even if the sections of requirements engineering, and monitoring the project progress are important to the acquirer, the SA-CMM are not very detailed in these areas. These areas should be exemplified to be of real use for the acquirer.

3.4 IEEE Recommended Practice for Software Acquisition

3.4.1 Description

This standard [3] is developed by IEEE[1] and was released in 1998. IEEE objectives with the standard are to:

- Promote consistency within organizations in acquiring third-party software from software suppliers.
- Provide useful practices on including quality considerations during acquisition planning.

1. The Institute of Electrical and Electronics Engineers, Inc., helps advance global prosperity by promoting the engineering process of creating, developing, integrating, sharing, and applying knowledge about electrical and information technologies and sciences.

- Provide useful practices on evaluating and qualifying supplier capabilities to meet user requirements.
- Provide useful practices on evaluating and qualifying supplier software.
- Assist individuals or organizations judging the quality of supplier software for referral to end-users.

The software acquisition lifecycle represents the period of time that begins with the decision to acquire a software product and it ends when the product no longer is used. The lifecycle represents a framework for acquisitions. It comprises five phases. The boundaries of the phases are defined as milestones for documents or important actions to be carried out. The output from one phase acts as input for another. The phases are illustrated below in Table 2.

Phase	Phases initiation milestone	Phases completion milestone	Steps in software acquisition process
Planning	Idea is developed	Release the RFP	1) Planning organizational strategy 2) Implementing organization's process 3) Determining the software requirements
Contracting	Request For Proposal is released	Sign the contract	4) Identify potential suppliers 5) Preparing contract requirements 6) Evaluating proposals and selecting the supplier
Product implementation	Contract is signed	Receive the software product	7) Managing supplier performance
Product requirements	Software product is received	Accept the product	8) Accepting the software
Follow-on	Software product is accepted	Product is no longer in use	9) Using the software

TABLE 2. Phases in software acquisition according to IEEE

The RPSA defines nine steps that should be accomplished, see Table 2. These steps should assure that products with potential for high quality gain maximum support from the acquisition process. The result expected should be a high-quality,

well-documented product. Quality attributes such as on-time delivery and cost-effectiveness are left for implementation by the reader. Table 2 shows phases, milestones and steps in each phase to accomplish.

The steps in the practice are intended for software that is bought with a basic set of functionality with possibilities to build on or software components that are completely finished. The steps are less suitable for software that needs to be built from scratch.

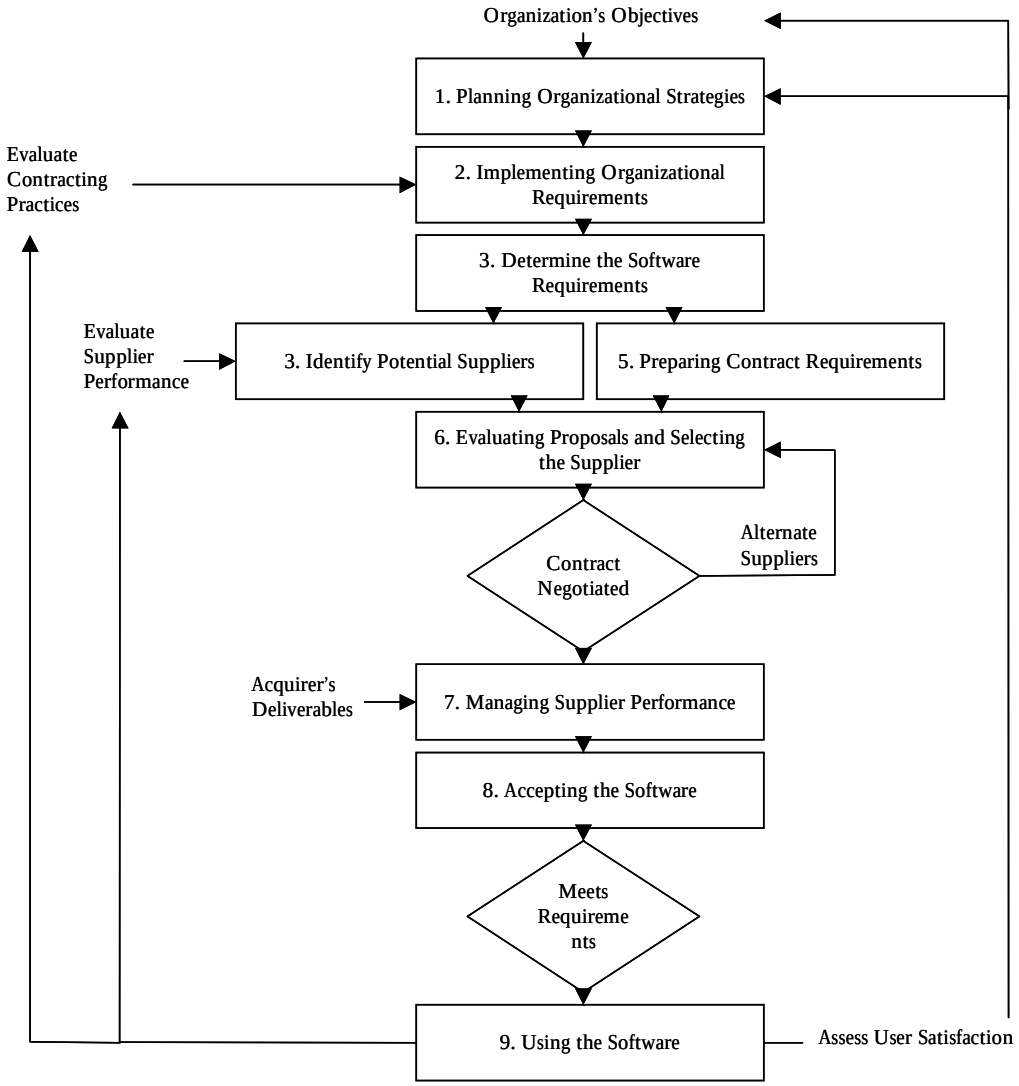

FIGURE 4. Steps for acquiring software

Step one in Figure 4 is *Planning organizational strategies* Review acquirer's objectives and develop a strategy for acquiring software.

Implementing organization's process Establish a software acquisition process that fits the needs of the organization for obtaining a quality product.

Determining the software requirements Define the software being acquired and prepare quality and maintenance plans for accepting the software.

Identifying potential suppliers Select potential candidates who will provide documentation for their software, demonstrate their software and provide formal proposals. Failure to perform any of these actions is basis to rejected a potential supplier.

Preparing contract requirements Describe the quality of the work to be done in terms of acceptable performance and acceptance criteria and prepare contract provisions that tie payments to deliverables.

Evaluating proposals and selecting supplier Evaluate supplier proposals, select a qualified supplier and negotiate the contract.

Managing the supplier performance Monitoring supplier's progress to ensure all milestones are met and to approve work segments. Provide all acquirer deliverables to the supplier when required.

Accepting the software Testing should be performed as well as establishing a process that ensures that all the discrepancies have been corrected and that all acceptance criteria have been satisfied.

Using the software Conduct a follow-up analysis of the software acquisition contract to evaluate contracting practices, record the lessons learnt and evaluate user satisfaction with the product. Retain supplier performance data.

3.5 Application of the RPSA to small-scale information systems

3.5.1 Planning

The planning process is initiated by developing a scope for the planning process, forming a planning group and identifying the qualities a software product must possess to achieve the organizations objectives.

IEEE suggests that the quality characteristics of the software should be determined before determining strategies for the acquisition. This is on the contrary with [20] "Quality Requirements for Software Acquisition" which reports that quality requirements should be measurable on the software product developed and hence they cannot be determined before you know what product to acquire.

The elicitation, weighting and specification of objectives should make the planning group aware of the comprehensive functionality and complexity of the small-scale information system that is needed to support the objectives.

IEEE provides a checklist [3] to assist in determining strategies. The strategy areas that IEEE elaborates on are the following: training of personnel, documentation, testing and maintenance. A general practice for handling suppliers may be documented to achieve consistency in contracting with suppliers.

The acquirer should determine to what extent the supplier's organization is involved in producing a high-quality product. RPSA further reads that responsibilities should be determined for the acquirer and for the supplier already in the planning phase. But the RPSA does not give substance to the importance of contemplating how the own organization may contribute to the delivery of a high quality product. This should be sorted out before determining responsibilities for both parties.

3.5.2 Implementing organization's process

IEEE provides a general process for acquisition [3]. The organization should derive the most suitable process from the framework that IEEE provides. The process includes definition of the following steps: planning, implementing of process in organization, software requirements, identifying potential suppliers, preparing contract requirements, evaluating proposals, managing supplier performance, accepting and using software. In reference to the composition of the organizational specific process the acquirer should prepare contracting practices and assign responsibility for success.

- Responsibility for success includes specifying technical performance and quality requirements.
- Managing supplier performance under the contract.
- Assessing supplier performance during the period of the contract.
- Evaluating and accepting the product.

3.5.3 Defining software requirements

The objective of defining the software being acquired is to obtain realistic assessment of size, scope and cost of the effort of developing the software from the supplier. If the software is defined really well, the accuracy of the suppliers' estimates increases.

The criteria for evaluation of suppliers should be specified. Specifying them before finding suppliers elicits the true needs and expectations. The obligations of acquirer and supplier should be established. Plans should be developed to evaluate and accept software and services. Contingency plans should be developed for use in the event that the supplier fails to satisfy requirements and the contract is terminated.

Requirements definition in this stage serves the purpose of obtaining realistic assessments of size, scope and cost of the development effort from the supplier. The software, deliverables and software support should be described as completely as possible. Research indicates that the acquirer has much to gain in defining functionality in form of scenarios [13].

3.5.4 Identifying potential suppliers

This should be done using the software requirements. Information about supplier may be obtained from sources as trade publications, consultants, suppliers and user groups. A potential supplier should demonstrate its products. The acquirer may benefit from an evaluation of the products from current users.

3.5.5 Preparing contract requirements

These requirements should state which obligations the acquirer and supplier respectively have towards each other.

- Define what constitutes satisfactory performance by the supplier, who is authorized to change in the contract, specify contract means for monitoring and identifying performance as well as functional specifications.
- Determine how payment is to be made.
- Determine what countermeasures the acquirer may use to constrain damage due to suppliers' inability to meet schedule or budget.

These requirements are important tools for the acquirer when it comes to spurring the supplier to not lose pace and to make sure that the project is easy to grasp. The definition of milestones could be combined with part payments. At the milestones, performance may be measured. For example as the number of requirements implemented and tested.

3.5.6 Evaluating proposals and selecting supplier

The evaluation criteria established should be used to review suppliers' responsiveness to the software requirements, deliverables and software support requirements, all described in the request for proposal.

It is a good idea to visit suppliers' facilities and evaluate factors such as financial position, technical capability, experience and quality practices. Then select a worthy supplier and negotiate the contract.

3.5.7 Managing supplier performance

The evaluation criteria established should be used to review suppliers' responsiveness to the software requirements and deliverables. The RPSA has not much to say when it comes to defining the criteria for monitoring of supplier's progress.

However progress in a software project can be made tangible. If the acquirer is concerned with optimizing for product functionality it is important to make the process of the development visible for the acquirer. If the supplier and the acquirer interpret the specified functionality differently, it is discovered early. The representation of software in the life cycle of the acquisition varies. Software is generally represented as scenarios, requirements, high-level design and low-level design. The progress monitoring is indeed dependent of the process used to develop the software. It is hence important to know which outputs to demand from the supplier.

- Managers should create an environment that supports the supplier's efforts.
- An individual should be appointed to deal with the supplier on all aspects of the contract.
- An open line of communication should be maintained with the supplier.

The acquirer should also monitor the supplier's progress to ensure that all milestones are met and to approve work segments:

- Use measures of reliability and quality specified in the contract to evaluate the supplier's work.
- Provide means for regular and continuous feedback on supplier performance.

All of the paragraphs need to be derived with examples to make them useful for the acquirer of a small-scale information system. They are important but hard to exemplify since they could vary a lot depending on the process used to develop the software.

3.5.8 Accepting the software

The software should be evaluated and tested compared to the acceptance criteria. The purpose of the test should be to detect discrepancies between existing and required conditions. The final acceptance test should involve field testing results.

3.5.9 Using the software

The acquirer should evaluate itself as a part of the acquisition process. Practices may be identified as weak and with need to be changed. Did any practice produce particular good results? Evaluate user satisfaction and record actual amount of maintenance.

3.5.10 Summary

The nine steps in the RPSA framework are generic and should apply to an acquisition of a small-scale information system if a special instruction was derived. However, each step involves choosing from many possibilities. The layman should appreciate fewer options to choose from. The paragraphs below describes what has to be modified:

- The standard is very comprehensive and could be used by very large companies without problems. On the contrary, a small company could not use it without modifications.
- The standards are hard to interpret. The reader must be experienced in the area of software engineering to understand reasons for certain procedures in the standard.

- IEEE focuses on the delivery of a high-quality, well-documented product [3]. On-time delivery and cost-effectiveness are left to be improved through the application of quality principles.
- Even though focus is on the delivery of a high-quality product the text and checklists in this area are not elaborate enough for an inexperienced acquirer.
- It states what is needed, not why, neither is any further information on how to implement the necessary features given.
- It is not specific in the area of monitoring progress. The one-shot acquirer needs more information than just the fact that the project should be monitored.

CHAPTER 4

Software processes

A process is a set of activities and associated results, which leads to the production of a software product [7]. The steps may vary but some fundamental activities are always represented:

- *Software Requirements Specification* The functionality of the software and constraints on its operation.
- *Software Design and Implementation* The software that meets the specification
- *Software Testing* Validation of software to ensure it does what the customer wants.
- *Software Maintenance* The software must evolve to maintain useful.

The acquirer's primary interest except for knowing about the process is the output of each step and what is demanded from the acquirer to make the process work. Which demands should the acquirer have on process and product deliverables? It depends among other things on the purpose of the acquisition.

This chapter will present common types of software process models. Then two instances of the abstract software process models are presented. The two phases of a process model: requirements engineering and verification and validation will be explained in depth.

4.1 The waterfall model

The model reflects good engineering practice. The waterfall model takes the four basic process activities and represents them as separate process phases. When a stage is completed the next one is adressed. The model maps into the following steps:

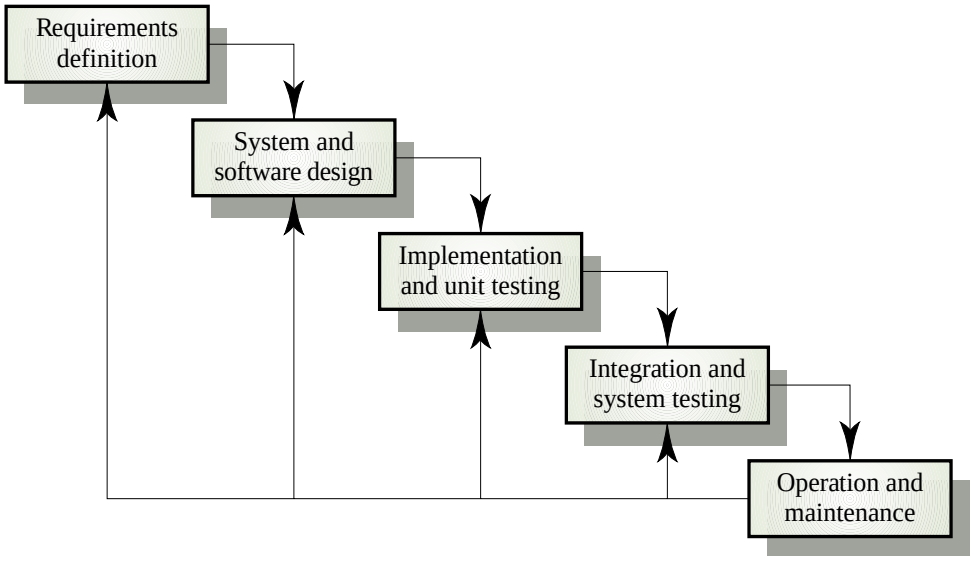

FIGURE 5. Waterfall model phases, see [7]

The phases in Figure 5 is described below:

- *Requirements definition* The goals are established and system constraints are composed in consultation with system users. The output from this phase should be a detailed system specification.
- *System and software design* In this phase the requirements are transformed into fundamental software abstractions and their relationships. The output is a high-level design document.

- *Implementation and unit testing* The high-level design is realized as a program units. These units are verified to meet its specification.
- *Integration and system testing* The units are integrated and verified to function as a complete system according to the requirements. This phase's output is the delivery of the complete system.
- *Operation and maintenance* The system is installed and put into practical use.

The aim should be to make things right the first time. The process does not allow iteration of the steps in the model. A phase cannot commence before documents in the current phase are satisfactory. The result of each phase is one or more documents that are frozen in its current condition. Problems are worked around or left for later resolution. The process is well suited for large projects with well known requirements. However, it is inflexible and it locks the projects into the distinct stages with baselines. This makes it hard to respond to changed customer requirements.

4.2 Evolutionary development

This model's basic thought is to commence with an initial prototype and refine it until the adequate system has been developed. This is illustrated in Figure 6. This is performed by exploratory development or throwaway prototyping.

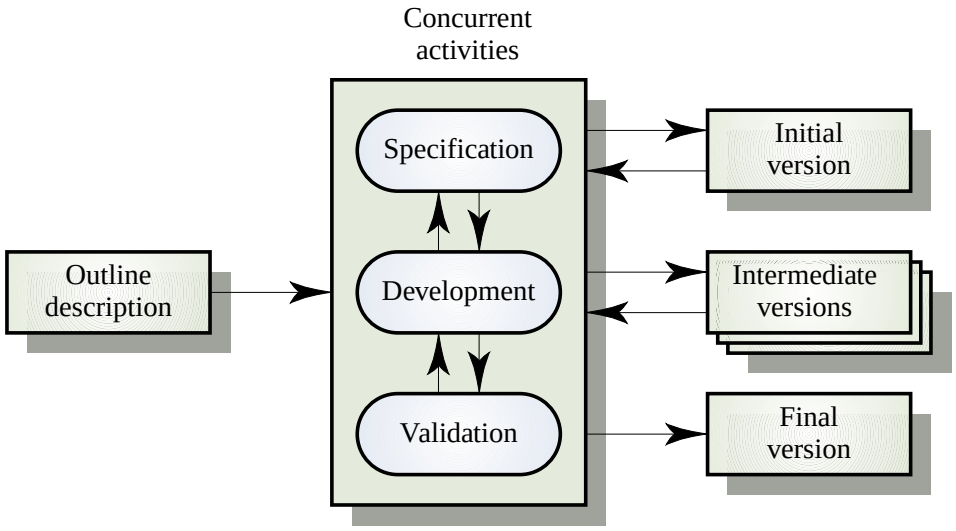

FIGURE 6. Evolutionary model concepts, see [7]

Exploratory development starts with the well known requirements and the developer works close to the customer to explore requirements. The purpose with the prototype is to deliver a working system.

Throw-away prototyping concentrates on the requirements that are poorly understood and experiments to understand them. Some requirements may not need to be modeled since they are understood.

The customer may expect the following advantages when acquiring software developed with this process approach:

- With this model the immediate needs of the customer is met.
- The development is rapid since it is not cost effective to produce documents that reflect every version of the system.
- Since the specification is developed incrementally users develop better understandability of the problem and this is immediately reflected in the requirements specification.

The following disadvantages should be considered:

- The architecture is specialized on the current version of the program. This means that it is expensive to maintain and to add features. This makes the system poorly structured.
- The development is rapid and it is not cost effective to produce deliverables that reflects every version of the system. The progress of the project using this process model is not visible. It is hard to measure progress and to keep track of the development.

This process is suited for small projects or parts of larger systems according to [7]. An implementation of this model is eXtreme Programming in Section 4.5.

4.3 Incremental development

This fashion of development combines features of the waterfall model with features of the evolutionary development model. An advantage with the waterfall model is that it is a simple management model and its separation of design and implementation leads to robust systems that are amenable to change. The evolutionary approach allows requirements and design decisions to be delayed but also leads to software that may be poorly structured.

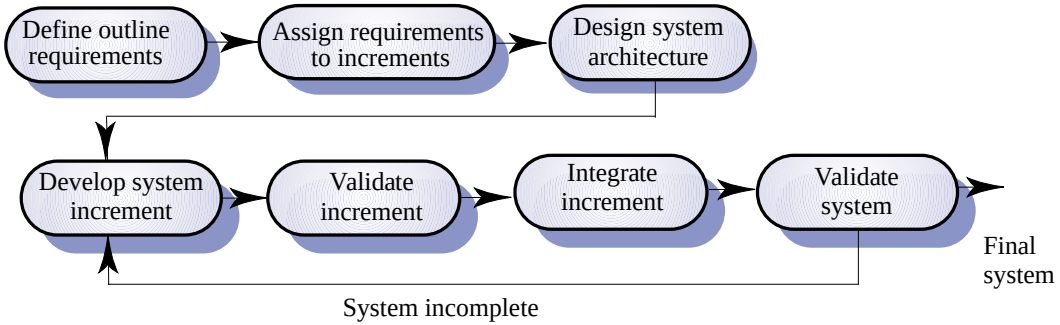

FIGURE 7. Steps of incremental development, see [7]

Figure 7 illustrated the steps of incremental development. The steps are described below:

- *Define outline requirements* Requirements are defined in outline for the system. Details are defined later when the system is partitioned into increments.
- *Assign requirements to increments* The system is partitioned into increments. Requirements for each increment are defined in detail.
- *Design system architecture* Requirements that are common for all parts of the system lay the foundation for the design.
- *Develop system increment* Increments that are prioritized by the customer are developed first.
- *Validate increment* Increments are verified to comply with requirements.
- *Integrate increment* The increments are integrated into the architecture to form the system.
- *Validate system* The increments are validated to function as a complete system.

The customer does not have to wait until the entire system is delivered to gain from its functionality. The customer may use the early increments as a form of prototypes and gain experience that helps to form requirements for the remaining of the system. The risk of the project to fail reduces since certain functionality will be delivered. The highest priority services are delivered first, which is good because this means that they receive the most testing. It is difficult to identify the parts of the code that affect all increment so architecture may suffer from this approach.

4.4 Spiral development

This model was originally proposed by Boehm in 1988, see [12]. The spiral development process is represented as a spiral rather than as a sequence of activities with backtracking. Each loop in the spiral represents a phase in the process. The phases in the process are not fixed, loops in the spiral are chosen depending on what is required. This process model considers risks opposed to other process models. Risks are explicitly assessed and resolved throughout the process in each iteration.

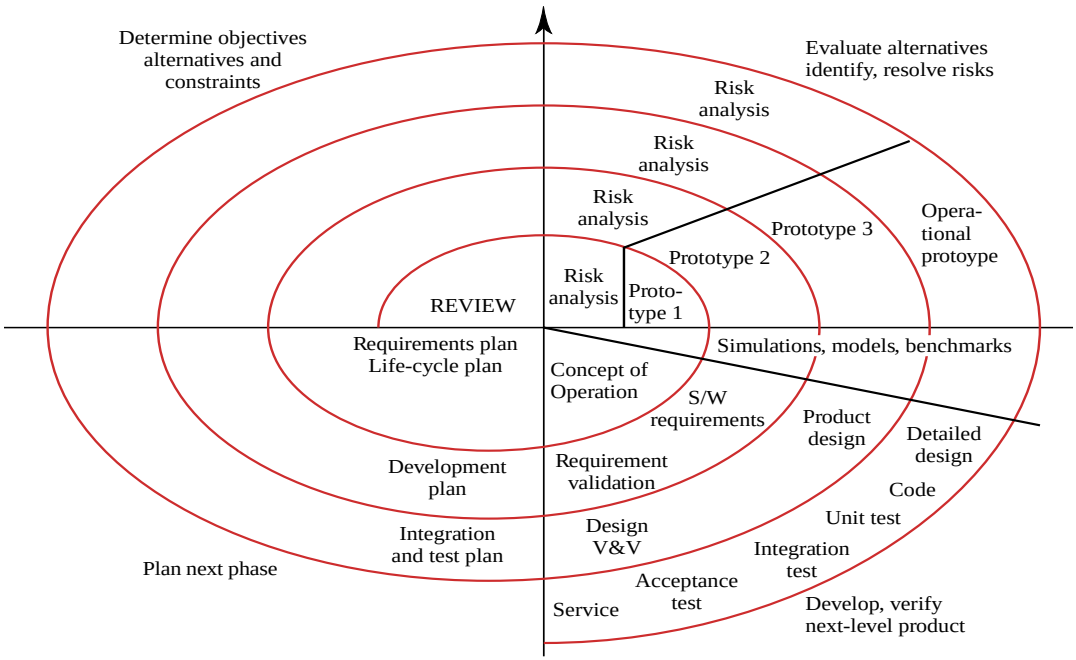

FIGURE 8. Boehm's spiral model of the software process (copyright 1988 IEEE)

- *In the objective setting* specific objectives for the phase are identified.
- *In risk assessment and reduction* risks are assessed and activities put in place to reduce the key risks
- *In development and validation* a development model for the system is chosen which can be any of the generic models
- *In planning* the project is reviewed and the next phase of the spiral is planned

An implementation of this process model is the Rational Unified Process described in Section 4.6.

4.5 Extreme programming (XP)

eXtreme Programming (XP) is a software development discipline developed by Kent Beck in 1996 [8]. This discipline uses the evolutionary fashion described in Section 4.2.

4.5.1 Customer communication

XP aims to keep a high level of communication through unit testing, pair programming and task estimation. The on-site customer decides what will be built and in what order. The effect of testing, paring and estimating is that programmers, customers and managers have to communicate.

A customer representative is recommended to be available for programmers ready to answer questions about the system. This makes for faster development of the system and helps programmers to understand the problem domain. It is of course expensive for the customer to make employees available for this activity. The gain is that the system may be released faster and needs less tuning once deployed.

Extreme programming puts greats pressure on the customer. Extreme programming suggests the customer to write stories[1] about daily tasks to programmers. The stories quality should increase as the programmer gives fast feedback to the customer on what information is considered to be important to the programmer. Understanding the customer problem domain is delegated to the customer in the sense that the customer should find the relevant information to provide to the programmer. This implies that the customer should have advanced technical skills to be able to elicit the relevant information.

This way of working means that the customer is able to understand the tasks that the programmer is working with. When developing a transaction based small-scale information system, programmers have to program an architectural framework. Having the framework as a common base to work on the supplier is able to implement features for the customer. Having an acquirer representative working with developers for understanding these parts of the system does not bring anything to developers or the acquirer. The customer being available for early support to programmers does not guarantee quality.

1. Stories are descriptions of task that the customer performs. The purpose of a story should be to describe the task in a manner that a developer could elicit the necessary information to implement the corresponding functionality in the system.

It is also essential that the customer representative is the right person to answer the specific questions about the scenarios that the programmers are implementing. Sometimes questions involve strategic decisions that cannot be settled by the customer representative. This may not be obvious when the customer representative authorizes certain actions. This may lead to much slower development, as management must review decisions anyway. Organizational schemes with communication paths should be established. Authorities for decisions should also be established and distributed. This makes sure that the right people makes the right decisions and that developers knows who to contact for which decisions.

When working in small cycles misunderstandings are solved relatively fast. However if this collaboration is going to work it is important that the right representatives are used for communication. Extreme programming [8] defines that the best customers are:

> the ones that are going to use the system being developed but also have a certain perspective on the problem to solve.

Users that are going to provide input to the system may not have the comprehensive picture and organizational domain knowledge. Hence representatives in the form of domain experts and users have to be utilized in different contexts. Domain experts' skills may be more adequate to determine if requirements are relevant and user skills may be used for testing look and feel of the system. Requirements that affect the architecture of the system are not likely to be decided on by the programmer, rather by the system expert leading to an economic decision taken by acquirer and supplier management. Representatives from acquirer and supplier are illustrated in Figure 9.

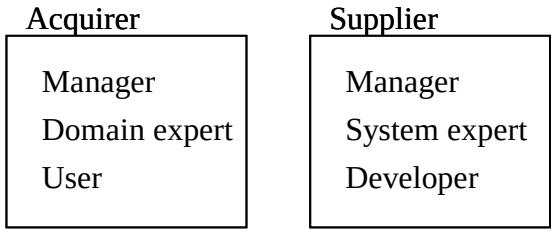

FIGURE 9. Representatives from acquirer and supplier

Further Extreme programming requires the customer to learn how to write functional test cases. The effort versus gain of delegating this task to the customer should carefully be considered. The approach of extreme programming its original form may not be cost effective for the acquirer defined in Section 1.4.

4.6 Rational Unified Process (RUP)

The rational unified process is an implementation of the spiral model in Section 5.4. The rational unified process are described in [28] as a process product, developed and maintained by Rational Software.

The RUP emphasizes the adoption of certain best practices, as a way to reduce the risk inherent in the development of new software. These best practices are:

- Develop iteratively
- Manage requirements
- Use component-based architectures
- Model visually
- Continuously verify quality
- Control change

The Rational Unified Process weaves these best practices into the definitions of:

- *Roles* as sets of activities performed and artifacts owned
- *Disciplines* for focusing areas of software engineering effort such as requirements, design, implementation, and test.
- *Activities* are definitions of the way artifacts are produced and evaluated
- *Artifacts* are the work products used, produced or modified in the performance of activities

The RUP is an iterative process that identifies four phases of any software development project. Over time, the project goes through inception, elaboration, construction, and transition phases. Each phase contains one or more iterations in

which you produce an executable, but possibly incomplete, system. During every iteration, you perform activities from several disciplines in varying levels of detail.

4.6.1 Advantages for the customer

With an iterative development approach it is easier to handle change of requirements. Requirements are likely to change during the project as needs become clearer as businesses change. With incremental development the system is built progressively. This eliminates the risk with integration at the end of the project. This risk is divided into smaller risks.

The customer decides in outline the services that are going to be provided by the system. They also decide on the most important and least important features. The system is now partitioned into subsets containing certain features. The requirements are defined in detail for the increment delivered first. The increments should be able to be put into service as soon they are delivered. Advantages are:

- Customer does not have to wait until the entire system is delivered to gain from its functionality.
- Customer may use the early increments as a form of prototypes and gain experience that helps to form requirements for later projects.
- The risk of the project to fail reduces since certain functionality will be delivered. They problems may be specific for a certain service.
- The highest priority services are delivered first. This means that they receive the most testing since they are the basis.

Iterative development opens the possibility for management to make tactical changes to the product. Short iterations make it easy to plan and estimate usage of resources. Another advantage with incremental development is that tasks that have to wait for the system to become complete could start early. Activities such as documentation and testing could commence as increments are being covered. The customer hence has a tested and documented prototype available all the time.

It is on the contrary difficult to map the functionality to properly sized increments. It is also difficult to identify the requirements that affect all increments since the requirements of the increments are not defined until the increments are going to be implemented.

4.7 Requirements engineering

It is important to know about requirements engineering since requirements may be used to select supplier, communication in project, measure supplier progress, to verify and validate functionality and so on. A requirements technique that pervades industrial practice is scenarios.

4.7.1 Scenarios

Although scenarios are widely used in the industry, studies on their practical relevance are rare. Surveys are mostly broad or draw their conclusions from a single project. The European Esprit project Crews are seeking a deeper understanding by comprehensive and expressive studies on the practical relevance of research techniques reported in Weidenhaupt et al. [13]. The study involved 15 European projects to learn how scenarios produced and utilised. They also identified benefits and problems associated with usage in industrial settings.

Scenarios are descriptions of tasks that users perform in their ordinary work. They are described from the perspective of the user. The scenarios do not have to include any interaction with a computer. They help the developers to understand the user domain and user tasks. The information in a scenario is easily provided by the end-users that are involved in the scenario.

One of the problems in Section 1.2.1 in the Lund case was that the acquirer had problems adapting to the technical jargon of the supplier. Scenarios may be used to concretize abstract models. Abstract models used by a developer may be hard to understand for the acquirer representative. The acquirer representative prefers to talk about concrete models rather than abstract models [13]. The other way around the business process to be supported may be hard to make out for the developers. In [13] comprehensive studies indicates that scenarios may be used for making it easier for the supplier and acquirer understand each other.

Scenarios promote interaction between supplier and acquirer. Scenarios require specific domain information that only domain experts can provide. Developers should use the language of the problem domain if helping to write scenarios. This is vital to establish good communication with non-technical experts. This is also the result of a case study performed by [14]. In [14] scenarios are recommended for use during acquisitions, since these types of requirements did not favor one supplier over another and helped to elicit other important requirements. In [14] the following advantages for the customer are mentioned:

- They may improve the understanding of the domain, omitting the need for a lot of detailed requirements.
- They make it easier to avoid premature design.
- They may easily include a lot of complexity and variants.
- Users find them easy to understand and validate.
- In some products they are sufficient precise to serve as requirements, in other they are excellent explanations of why various requirements are needed.

In [19] "Deriving Goals from a Use-Case Based Requirements Specification" scenarios are also proposed to be used by software engineers to gather and validate requirements.

Scenarios are not use-cases. Use cases are another style of expressing require- This style the developers may want to use to express the requirements of the product. These define how the system or application works. The use case defines actors, a brief description, and pre-conditions, the main flow, alternate and sub-flows and exception flows. The use cases may not be from a users perspective but may come in contact with flows in the software that has no contact with a user. These are therefore harder to validate for the user.

4.7.2 Coexistence with prototypes

Prototyping varies from simple paper-based user interface to a comprehensive system. Paper-based prototypes may be as useful as comprehensive systems, this is indicated in [22]. Prototyping serves not only as concretizing scenarios but also to verify the scenarios them selves. Scenarios may serve as to validate the initial prototypes and indirectly to verify the requirements specification.

Recent research in [18] "Preventing Requirements Defects" indicates that the number of usability defects may be reduced by 70% percent if scenarios are used in conjunction with early usability testing. Prototypes may be used in the initial stage for evaluation of suppliers or in the development process to concretize or complement scenario requirements. In [18] "Preventing Requirements Defects" scenarios are stressed as an important technique since it is useful for identifying the tasks to be used in usability tests. This apprehension is also adapted by [19] "Deriving Goals from a Use-Case Based Requirements Specification", they report that scenarios are used in HCI to improved communication between end-users and developers for

designing user interfaces, task modeling, prototyping and supporting the specification of user interfaces.

4.7.3 Quality requirements

In order to satisfy the quality objectives, requirements may be developed to build confidence in that the quality objectives are met. In some cases quality objectives may not be available for measurements until some maintenance has been performed on the product. Requirements may then be set on the development process used to fulfill the quality requirements. If the functionality of the product is important and may need further specification after selection of supplier, the acquirer should require that users a involved to a high degree in the development of the functional specification. Requirements on prototyping may also be helpful. Further if the acquisition objectives are accuracy of the output of the software, for example specific calculations, requirements on the testing of this functionality may be helpful.

Except for requirements on the process used to develop the product, requirements may be set on monitoring of the process to see how effectively the process is applied. Anther need for project monitoring requirements are that they are related to the stability of functional requirements. If functional requirements often change then it is necessary to require that communication processes between supplier and acquirer are strengthened, this is implied in [20]. If process visibility is important requirements may be set on traceability between phases, such as indicators of how many of the requirements are working features of the product. In testing demands may be on how many tests cases are passed, number of defects corrected and found.

Requirements could also be set on the acceptance of the product. Those could be set on the efficiency, usability, reliability, maintainability, interoperability, etc. But requirements could also concern traceability requirements that could be important for example when transferring knowledge in the acquiring organization.

4.8 Verification and validation

This testing requires a real user to interact with the end product by executing. The goal is to adjust the interface of the program to the users style of work, instead of

adapting the users to the program. The users are recommeded to be involved as early as possible by Edward Kit in [21]. This kind of evaluations should be focused on the presentation of the program rather that its functionality.

Usability characteristics which can be tested include the following:

- Accessibility
- Responsiveness
- Efficiency
- Comprehensibility

CHAPTER 5

Current practice of software acquisition

In order to find problems with the current practice of software acquisition two projects are investigated. The projects are acquisitions of small-scale information systems. Both projects were performed by the local authorities Lund and Malmö. Interviews with the project managers from both projects were performed, they are listed in Appendix A.

The interviews reveal several problem areas. The areas that problems are found in are:

- Communication
- Planning
- Requirements
- Verification and validation

These areas are explained in more detail below.

5.1 Communication

The two projects used for the study are dissimilar in the way communication with the supplier has taken place. The Malmö project has had intensive communication between end-users and developers, giving feedback on new releases on regular bases. This is opposed to the Lund case where project communication has been very intensive between the managers. The contact between developers and end-users has been rare.

5.1.1 Lund

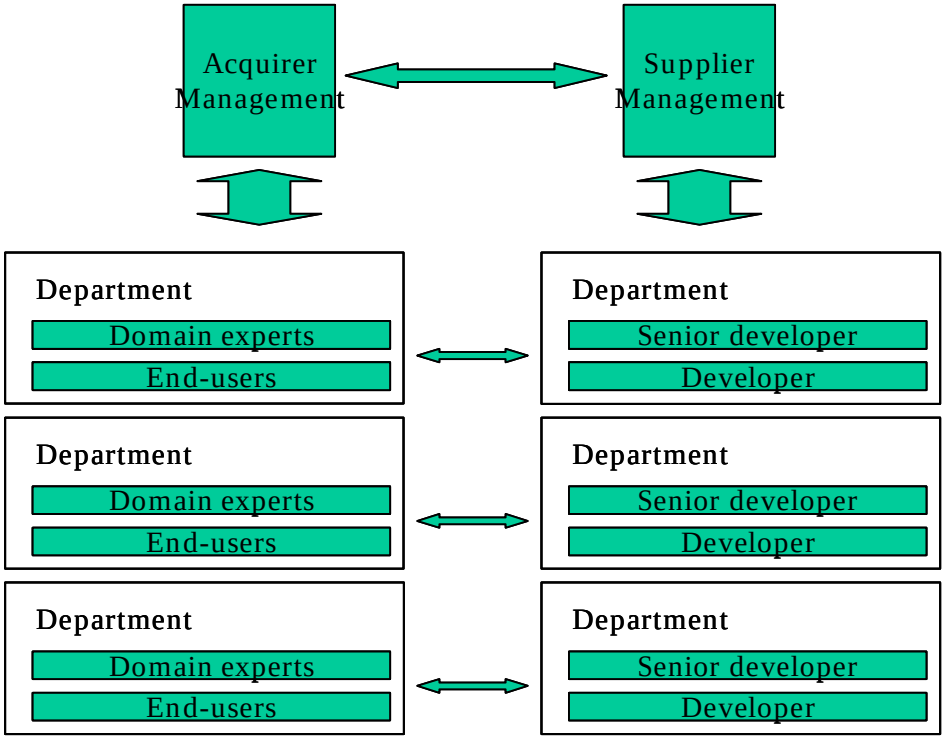

FIGURE 10. **Communication in the Lund case**

As displayed in Figure 10 almost all communication was directed trough the managers who had daily contact during the project. Meetings seldom took place between the department personnel and the developers. At meetings communication was problematic, since the acquirer and supplier had little previous experience of each others domains. The management experienced that the communication with the supplier was excellent.

5.1.2 Malmö

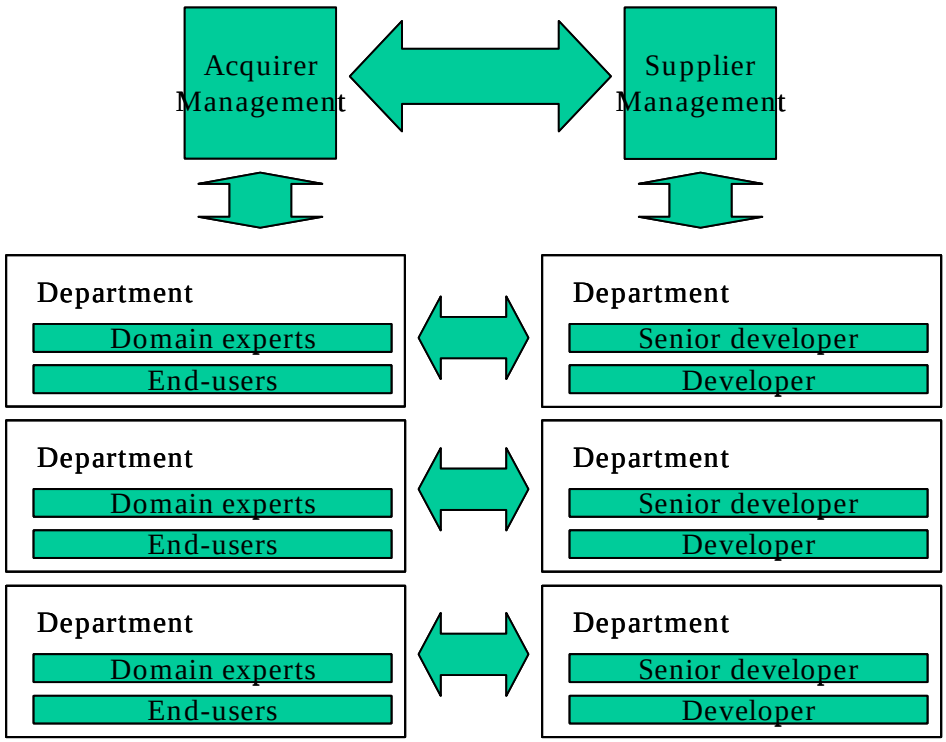

FIGURE 11. Communication in the Malmö case

The intensive end-user involvement that the Malmö case had made sure that the product was tweaked to the context by the future users. The communication is pre-

sented in Figure 11. This type of communication promoted the acceptance of the product. It was expensive to let the end-users off from their regular tasks to work on the project. Neither of the projects has considered the cost of end-user involvement in addition to the price of the product acquired. This could be really useful if the costs/savings of introduction of an information system in an organization should be calculated.

Even though the projects have had dissimilar communication, both acquirers experience that the relation has been very good with the supplier. The Malmö project has had thoroughly end-user involvement and the Lund project had not. However both projects were more expensive than planned for and has gone beyond the planned deployment date.

5.2 Planning

Neither of the projects had a time plan that related to the complexity of the software. The projects compiled the time plan related to stakeholders interests in releasing the information systems to the public. Neither of the interviewees are able to motivate the estimations in their time plans. The time plans do not seem to have any correspondence with the complexity of the requirements of the system. When the time plan is created from the acquirer's business point of view, the requirements should be reviewed by the supplier with suggestions of the time it takes for implementation. Then prioritization of requirements will most likely take place as the business planning probably does not correlate with the time for implementation of requirements.

5.3 Requirements

The two acquisitions also have in common that the managers have experienced an "uncontrollable stream of requirements during the project". Requirements that were elicited until the time for the selection of supplier were not enough when the projects started.

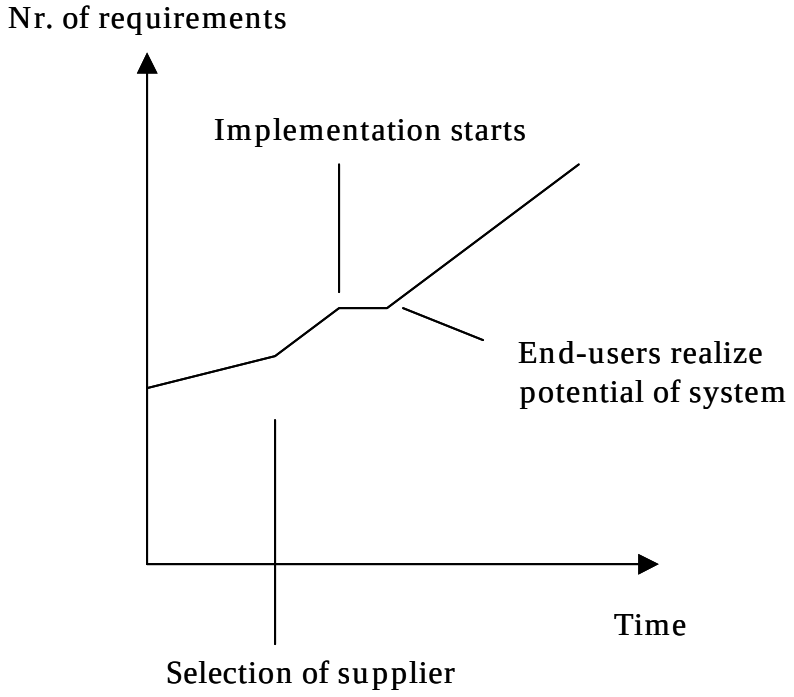

FIGURE 12. End users realizes the potential of the system at a late stage

New requirements cropped up as soon as the projects began. The suppliers had from the beginning of the projects an increased workload compared to the workload that was planned for. After a few meetings the business perspective of the time plan had not changed but the requirements were not prioritized and the workload was increased. In the Lund project, even two years later, new requirements are elicited by users that realize the potential of the information system. This phenomenon is illustrated in Figure 12. The angles of inclination in Figure 12 are fictitious.

The time plan for one of the projects was regularly updated. Since a stream of new requirements existed, the time plan followed the project instead of the other way around. The total time for requirements implementation increased continuously at meetings with the customer.

The project also experienced that the supplier was slow to fix problems after evaluations. This probably depended on the fact that the supplier had a lot more workload than expected from the start. Little time was left to work with the quality of the features since new features always waited to be implemented.

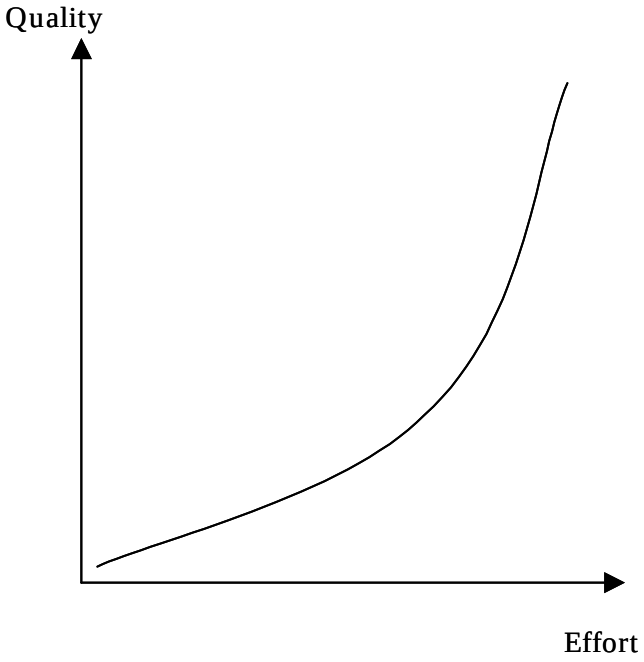

FIGURE 13. Quality of a feature does not scale linear if an even effort of work is performed

Quality of features does not start to evolve just as the features are implemented but after a while of working with it, Sommerville [7]. This is pointed out in Figure 13.

The new requirements were sometimes, but not always documented in the projects. During the projects, managers could not say if 200 or 300 new requirements had been elicited from the project start. The Lund case did have an overview of requirements that the Malmö case did not have, since everything passed through the managers in the Lund case. In the Malmö case requirements could more easily

be passed on to a developer without knowledge of management or being properly documented at meetings.

5.4 Verification and validation

The evaluations of the software product were conducted in different manners. The Lund case was evaluated during a month's trial period when intensive bug reporting and correctives actions were taken. The Malmö project performed evaluations of the software with the end users. This promoted the acceptance of the software but also generated a lot of overhead communication. E.g. three tests might give three different results if performed in three different departments. The involvement of the end-user however is not a bad idea if the results of the tests are agreed on before contact with the supplier.

Suppose that the supplier generates an easy to follow manual for test-cases intended to be used by the end-users. The tests are already tested to work. The end-users test them again and then get a look and feel of the interface of the software. The end-users will probably have opinions on the interface. Except for the tweaking of the interface this promotes quality since the tests are run several times with the real end-users. This type of testing is also a way for the supplier to show the customer that the software is working as agreed on. The test cases, if written as in a tutorial manner, could easily be transformed into manuals.

5.5 Summary

The two projects described above are used for identifying problem areas in software acquisitions. The organizations both acquire small-scale information systems. They have similar organizations since they both are Swedish local authorities. The two projects are therefore utilized for showing the variety of problems areas that may exist, rather than for comparisons. The problem areas lie in the phases of planning, requirements elicitation and management, and verification and validation. Both projects would benefit from experienced management with knowledge in these areas. The projects would also benefit from educating the end users to understand

the demands of this kind of projects. The improvement areas are the basis for the recommendations in Chapter 6 were they are used in conjunction with standards from Chapter 3.

CHAPTER 6

Guide to important areas for acquiring small-scale information systems

As described in the introduction, acquisition projects often fail. The projects are not unique but similar, and in many cases it could be figured out by software engineers where and why the projects took a wrong turn. Why does projects still fail even 20 years after the notorious CMM was introduced?

Such comprehensive guidelines were never compiled with the acquirer characterized in this thesis in mind. Even though support exists for projects today, it is certainly not easy to make use of. Therefore a pragmatic guidance to important areas may help the one-shot inexperienced acquirer.

The examination of standards in conjunction with interviews is used to guide the software acquirer to areas of importance when contemplating improvement of the acquisition process. The purpose is not to present an as complete list as possible, but to exemplify countermeasures for several of the problems discovered in the standards and the interviews and addressed in the standards.

The areas that are identified to be problems areas should management put effort into controlling:

- Strategies and planning
- Requirements elicitation and management
- Verification and validation
- Maintenance

The standards in Chapter 3 have recommendations for each of the areas above. Below the recommendations from the standards are described. These recommendations may be enough for handling the problem areas in Chapter 5.

6.1 Strategies for acquisition

The strategies should permeate the acquisition from the beginning to the end. The strategies give the acquirer a basis for making decisions throughout the project. The strategies should also form the boundaries of the software requirements specification. The purpose of the software acquisition should be clear for all persons involved since it is likely that stakeholders of the software may have conflicting views of the purpose of the software acquisition. A prioritization among stakeholders' objectives considering time, cost and functionality should solve this. Then the criteria for success should be determined. The criteria should consist of the most important objectives since all objectives may not be met.

The strategies are proposed to be documented as follows, according to SA-CMM [6]:

- Definition of objectives of the acquisition.
- Definition of project constraints, such as funding and schedules.
- Definition of software acquisition methods.
- Definition of potential contract types and terms.
- Definition of risk identification.

In Chapter 5 it was described that the different departments had different requirements on the software. Making strategies and prioritizing the objectives of the acquisition could solve this conflicting view of what the software should perform.

The area of risk identification is important for all acquisition projects. It may be hard for the one-shot acquirer to predict some of the common risks. Below are some common risks. Recommendations for avoiding the risks may be found in Department of Defense's "Guidelines for successful acquisition and management of soft-

ware-intensive systems" [29]. Management should know how to prevent these risks by defining possible countermeasures:

- Definition of countermeasures to the risk of unrealistic estimates of cost, schedule or size.
- Definition of countermeasures to the risk of cost of development exceeds any benefits the system may offer during its useful life.
- Definition of countermeasures to the risk of schedule is out of control.
- Definition of countermeasures to the risk that the software system does not perform as originally intended or thus fails to meet requirements.
- Definition of countermeasures to the risk of a software product that is so poorly constructed or complex, that it is too costly or impossible to upgrade or maintain.
- Definition of countermeasures to the risk of the project adheres to suppliers' problems and poor practices.
- Definition of countermeasures to the risk of requirements being out of the management's control.
- Definition of countermeasures to the risk of an unreliable product where quality levels are poor to marginal.
- Definition of countermeasures to the risk of buying a product that is very hard to use.

6.2 Planning for acquisition

When the strategies are thought of, an idea of the characteristics of the software should evolve. Information from stakeholders can be obtained e.g. by questionnaires or by interviews. The contemplation of the characteristics of the software will help to form the requirements. According to SA-CMM [6] the project plan should contain these paragraphs:

- Definition of problem statements.
- Definition of solution statements (definition of success).

- Definition of quality objectives.
- Definition of prioritization of needs.
- Definition of roles for project members.
- Definition of the project scope.
- The software parts of the project should also be defined and included in the document:
  - Risk identification.
  - Management.
  - Solicitation.
  - Requirements development management.
  - Evaluation and transition support.
- Definition of the tasks to be performed.
- Definition of the required resources including funding, staff, equipment and tools.
- Definition of how change should be managed (requirements elicited during the project etc.).
- Definition of master schedule of acquisition milestones.
- Definition of measurement to determine the progress of the acquisition.

The planning document is comprehensive and as described in SA-CMM [6] the reason for this is that the costs of the acquisition do not end as the supplier is selected or as the software is fully developed but when the software product are replaced or no longer used.

In Chapter 5 one of the projects did not have criteria for knowing when to stop developing the software. They did not have a strong defined solution statement.

6.2.1 Planning for prioritization

Both projects in Chapter 5 did have problems with a growing list of requirements during the entire projects. When this is the case, prioritization must be done. In a report by the Standish Group International [27] it is claimed that in many cases, 20% of a project's features will provide 80% of user benefits. Therefore when the project is slow or costs too much, it is good to know what to focus on.

6.2.2 Planning for roles

In the Malmö project in Chapter 5 the management felt that the end-users had poorly defined project roles. Definition of roles for project members is very important so the project members know which authorities they have. They can then be certain of what actions they may take in different situations. In projects were contact between developers and end-user is tight definition of roles is important. These paragraphs should be dealt with according to Standish Group International in [27]:

- Definition of everyone's roles.
- Definition of the type of users that should be involved.
- Definition of schedules for user involvement.
- Definition of how user involvement could be promoted.

6.2.3 Planning for quality objectives

The projects in Malmö and Lund did not express any thoughts about the qualities of the software. Neither did they use quality as a term related to the functionality of the software. Below are some of the quality objectives from SA-CMM [6] that an acquirer should think about:

- Definition of usable documentation.
- Definition of adequate resources for the software support organization.
- Definition of the warranty of the software product.
- Definition of the software process capabilities.
- Definition of the life cycle costs and schedule estimates for the software.
- Definition of how the quality objectives should be verified. E.g. by a demonstration, user survey, test, documentation review.
- Definition of responsibilities for making the evaluation.

6.2.4 Planning for payments

The projects in Chapter 5 had good co-operation with the suppliers. This is not always the case and payments could be used as penalties for poor performance. It

may also be used as a spur. The acquirer should plan for the payments as a part of the projects milestones. Below some paragraphs are presented that would be helpful when contemplating payments, from SA-CMM [3] and RPSA [7]:

- Definition of the minimum amount to be paid before the quality of the supplier's work is demonstrated.
- Definition of deliverables, such as documents at milestones or test results, that verifies the product so that payments are made in relation to measurable achievements.
- Definitions of which requirements are the most important so that payment may be reduced if these are not met. Be prepared to take out the prioritizations of the requirements to decide if it could be neglected to a lower price to be able to move on.
- Definition of what a complete software product consists of. Except for reducing the price at certain milestones, be prepared to reduce the price at the end of the acquisition for the completeness of the product.

  Be prepared to give the supplier feedback on its performance.

6.3 Requirements

In Chapter 5, new requirements were introduced during the entire projects and in the Lund case even after the project was closed. Therefore it is important to elicit the real need for acquiring new software. When interviewing users of an existing information system their objectives to acquisition of the new system may reflect functionality and routines used by the already existing software information system, as in the Malmö case. It is therefore important for the investigators to find the real objectives and needs. This is not always the need expressed by users.

It is hard to meet all stakeholders needs and prioritization is needed to meet the most important ones and to scale the objectives to a rational level for the acquisition. Prioritizing between stakeholders' objectives considering time, cost and value of the new functionality provided. Then determine the final objectives.

6.3.1 Stakeholders

Stakeholders are any person or organization that has an interest in the result of the acquisition. It could be essential to be able to answer the questions whether a person or organization will be affected by the acquisition or not. For example for the purpose of integration with other information systems as in the Malmö case in Chapter 5. Common issues regarding stakeholder are addressed in "Software Acquisition: A Comparison of DoD and Commercial Practices" [1].

It is hard to meet all stakeholders needs and prioritization is needed to meet the most important needs and to scale the objectives to a rational level for the acquisition.

6.3.2 Requirements planning

To avoid misunderstandings between supplier and acquirer it may be a good idea to create a mutual understanding of the requirements before an agreement is signed. In Chapter 5, in the Malmö case an agreement was reached before the supplier realized much work it had agreed to perform to a certain price. This may affect the project in a negative way. When the mutual understanding exists, the requirements may be baselined to make sure that appreciations of the implementation time are built on the baselined requirements as a starting-point. A requirements plan should typically contain, according to SA-CMM [9]:

- Definition of the software technical requirements:
  - Functional requirements or use cases or scenarios
  - External interfaces
  - Performance requirements
  - Quality attributes (reliability, security, maintainability, usability, etc. se below in the checklist)
- Definition of the tasks that should be performed:
  - Evaluation tasks
  - Support tasks
  - Documentation tasks
  - Life cycle planning tasks
- The document should also contain the contract documentation:
  - Specifications
  - Test plans

Procedures and reports

Configuration management plan

The configuration management should at least define how new requirements are documented, compiled into baseline and who is responsible for approving new requirements. This had been useful in both projects in Chapter 5 even if the requirements were documented in one of them it could be done in a more formal way. This making the new requirements documents easy to read for all parties.

6.4 Verification and validation

The evaluations of software worked quite well in the projects in Chapter 5. Below are recommendations for further improvement. If evaluations are performed by the end-users, they should be documented in evaluation plans containing these paragraphs, as in SA-CMM [6]:

- Definition of the purpose of the plan.
- Definition of the design of the test protocols.
- Definition of communication and responsibilities.
- Definition of a framework for how to carry out tests.

6.5 Maintenance

Maintenance was not contemplated at all in the projects in Chapter 5. It should be since the acquirers do not buy software very often. Maintenance could be as much more expensive than development according to Sommerville [7]. If the software acquired is built for being in use the next five years, maintenance calculations should be part of the picture. The software should include documentation so that the acquiring organization has a possibility of managing maintenance. Presented

below are paragraphs supported by RPSA [3] that would be helpful when transition to maintenance is performed:

- Definition of who will provide the software support.
- Definition of personnel responsible for the maintenance/support of documentation during the acquisition project so the knowledge could be passed on.
- Definition of who owns the code after development.
- Definition of what is included in the support of the system.
- Definition of how correction actions will be performed and by who.
- Definition of who will perform modifications of the software.
- Definition of who will make updates of user documentation.

CHAPTER 7

Summary and further work

7.1 Practical methodology

This chapter is a summary of the thesis. It also contains pointers to further work.

7.1.1 Literature studies

The focus was initially to understand the disciplines involved in a software acquisition. Literature studies did not reveal much, as the area is poorly documented.

To understand which disciplines are interesting to investigate, several local authorities were contacted. The authorities that were contacted had recently performed acquisitions. An open interview with Malmö kommun gave a jargon and a context to work with. This lead to better understanding of the standards SA-CMM and RPSA and also to be able to sift other literature. The literature study revealed several issues:

- The standards for software acquisition are not written with the acquirer defined in this thesis in mind. However they comprise useful knowledge of the area. Their usage are foremost a checklists for what has to be done. Standards comprise what should be done, but not why and how it should be done. Since formality varies in organizations it might be more important to tell why the tasks

should be performed than how. If the reader is familiar with the "why" and the resulting document, then the "how" still may be excluded since it is likely to vary anyway.

- No silver bullet exists among software process models for the customer. The supplier should be responsible for controlling and knowing this area. Is it important to know what demands that the software process model has on the acquirer. More communications will cost time and effort and it will require skilled staff to communicate with the supplier. On the other hand, software acceptance may be higher and most of the problems are solved and no surprises wait at introduction in the organization. Less communication may lead to a higher acceptance threshold in the organization and a lot of user testing waits. If evolutionary development is used, documentation could not be expected to be of high quality. On the other hand the project will move really fast. Incremental style minimizes risks in the area of software delivery and delays. Detailed requirements decisions but architecture may suffer and may lead to high maintenance costs. Inputs and feedback to the processes and the supplier should be an easy task for a non-technical acquirer. Requirements may take the form of scenarios to make it easier for the acquirer and the feedback from the acquirer should be structured and uniform to make it easier for the supplier.

7.1.2 Interviews

Focused interviews were then performed. The first interviewee was with a consultant and a person working for Malmö kommun. Then an interview with personnel at Lund kommun was performed. Other interviews were also performed, e.g. with Kävlinge kommun, however these have not been used since they did not match the profile of the acquirer in this thesis. The Lund and Malmö project was then used to describe issues in software acquisition projects.

The interviews gave a picture of what practice is used today by experienced acquirer's. The interviews were too few to be able to see any indicia at all. The summary of the interviews rather focus on showing a diversity of problems. The problems exists in these areas:

- Communication
- Planning

- Requirements
- Verification and validation

The problems in these areas are also addressed in the standards in Chapter 3. Recommendations from the standards are used to exemplify a structured way of working with the areas above.

7.1.3 Result

In Chapter 1.2.2 "Issues for investigation" seven paragraphs from "ADB-Projekt" [5] were suggested to be investigated with additional five paragraphs derived from the Standish Group investigation [27]. Paragraphs one to seven are all addressed in the SA-CMM and the RPSA as paragraphs in several planning documents. Hence it is important to know about them and to control them. If they are the success factors that all projects could contribute from, this thesis could not answer, since no evaluations has been done.

The paragraphs from Chapter 1.2.2 that are left unanswered are:

- The implementation of non-existing support for acquisitions. No recommendations for the most appropriate practice are described.
- The wish for being able of optimizing for costs, schedule or functionality is left for further work.

Chapter 5 contains an examination of problem areas. These are used in Chapter 6 with suggestions to improvements from the standards. The reader of this thesis should now be able to contribute to the purpose of increased knowledge of the area of software acquisition to be able to improve one self, as described in Section 1.3 (Figure 2). The reader should know about some areas of improvement and might have ideas on what to improve.

7.2 Further work

For this thesis, more work could be done in these areas:

- The guide to areas of interest for the small-scale information systems acquirer may be evaluated in acquisition projects to see if the areas described are the most appropriate to improve.
- The guide to areas of interest for improving the acquisition process may be expanded to discover several areas of improvement not mentioned in this thesis.
- The empirical investigations are quite limited in this thesis. They may be expanded to comprehend more projects.
- If a quantitative approach is taken, the projects examined could be used for identifying ingredients in successful projects.
- Another area of interest to examine is the specified costs and the actual costs of an acquisition.

CHAPTER 8

References

[1] Software Acquisition: A Comparison of DoD and Commercial Practices, (CMU/SEI-94-SR-9), October 1994

[2] IEEE, Std 610-1990

[3] IEEE, Std 1062-1998 "Recommended Practice for Software Acquisition"

[4] Premiepensionsmyndigheten Pressmeddelande http://www.ppm.nu/nonbfiles/pdf/284.pdf

[5] Undall Björn, "IT-utvecklingen inom staten 1999", ISBN 9174983946

[6] Software Engineering Institute, “Software Acquisition Capability Maturity Model (SA-CMM^{sm}) Version 1.01”, 1996

[7] Sommerville I., "Software Engineering 6th edition", Addison-Wesley Publishing Company, ISBN 0-201-39815-X, 2000

[8] Beck Kent, "eXtreme Programming eXplained", ISBN 0-201-61641-6, 2000

[9]

[10] Humphrey Watts S, “Discipline for Software Engineering”, Addison-Wesley Publishing Company, 0-201-54610-8,1995

[11] Software Engineering Institute, “CMMI SM for Systems Engineering/Software Engineering/Integrated Product and Process Development/Acquisition, Version 1.02d”

[12] Boehm B. W., "A spiral development of software development and enhancement", IEEE Computer Issue 5,1988

[13] Weidenhaupt Klaus et al., "Scenarios in system development: current practice", IEEE software March/April, 1998

[14] Lauesen, Sören, "Software Requirements - Styles and techiques", Samfundslitteratur, ISBN 87-593-0794-3,2000

[15] Björnsson Örjan, "Kvalitetsrevision 1-4 för Malmö Stad av BUV-projektet", 2000

[16]

[17] Paulk M.C., Curtis B., Chrissis M.B., Weber C.V., “Capability Maturity Model for Software, Version 1.1”, CMU/SEI 93-TR-24, Software Engineering Institute, ISBN 0-201-54664-7, Feb 1993

[18] Lauesen Soren, Vinter Otto, "Preventing Requirements Defects: An Experiment in Process Improvement", Requirements Engineering Journal 6:37-50 Springer-Verlag, 2001

[19] Antón Annie, Carter Ryan, Dagnio Aldo, Dempster John, Siege Devon, "Deriving Goals from a Use-Case Based Requirements Specification", Requirements Eng 6:63-73 Springer-Verlag, 2001

[20] Robert Philippe, "Quality Requirements for Software Acquisition", IEEE, 1997

[21] Kit Edward, "Software testing in the real world: improving the process, Addison-Wesley, ISBN 0-201-87756-2, 1995

[22] Grady Helen, "Web Site Design: A Case Study in Usability Testing Using Paper Prototypes, 2000

[23] Brandt Kjeld, "Vetenskaplig metod", Studentlitteratur, ISBN 91-44-36612-4, 1993, 1996

[24] Lantz Annika, "Intervjumetodik", Studentlitteratur, ISBN 91-44-38131-X, 1993

[25] DePoy Elizabeth, Gitlin N. Laura, "Introduction to research", Mosby, ISBN 0-8016-6284-2, 1998

[26] Merton Robert, Fiske Marjorie, Kendall Patricia, "The focused interview", Free Press, ISBN 0-02-920985-4, 1990

[27] Standish Group International Inc., "The CHAOS study", http://standish-group.com/visitor/chaos.htm, 1995

[28] Kruchten, Philippe, "Rational Unified Process", Addison-Wesley, ISBN 0-201-70710-1, 2000

[29] Department of the air force - Software technology support center, "Guideline for Successful acquisition and management of software-intensive systems", May 2000

Appendix A

Listing of interviews

Below follows listings of the interviews performed at the local authorities Lund and Malmö.

A.1 Case study A

A.1.1 Description

Malmö is a city in Sweden with 250 thousand habitants. Malmö has an IT division dedicated to acquisition of software and in some cases maintenance of software. The division is funded by the city budget and is an interface for the city towards suppliers of software.

Recently the IT division acquired software for administration of school districts and transfer of students between various schools. The administrative information support was a web based three-layer application with a database in the bottom. The acquisition started in autumn 1998 and was transfered into maintenance august 2001. The project was budgeted to 7,4 milijon. When finished it had costed 8,9 miljons.

A.1.2 Interview characterization

Two interviews with employees at Malmö stad are performed. The first interview was an open interview following the respondent's thoughts, delimiting the subject to the interest of the interviewer. The interviewee was a full-time employee in the project management. I have chosen not to display this interview. The purpose of the interview was to gain understanding of the problem domain.

The basis for the seconds interview is the issues in Chapter 1.2.2. The second interview was with a third party software project quality inspector. The interviewee has had a clear insight in the project on all levels. The interviewee was an employee in the project working as a software engineering project expert trying to improve quality in the project.

A.1.3 Interview

What did the criteria for selection of supplier look like?

The applications were acquired with requirements describing short scenarios exemplifying the wanted functionality. Criteria for selection of the most appropriate supplier were established. The criteria were of various importance's for the acquirer. These criteria may be read below:

- Application correspondence with requirements 40%
- Technical platform 35%
- System requirements

  Technical environment
- The company 15%

  Suitability of supplier

  Project model, organization, resources and competence

  Time plans

  Control and follow-up

  Quality assurance

  References
- Investment 10%

  Price

  Maintenance / Support

License price

A final date for the project completion was also set. No further information between start and completion was set at this stadium. However it was a request from the acquirer that the supplier assesses the effort of developing the desired functionality.

Why did you select the supplier/bid that you did?

The supplier chosen had a price that was outstandingly low in comparison with the others. The supplier made it clear that it was a favor for the city. In return the company hoped for cooperation with the city in future projects.

Did you have an alternative plan if none of the suppliers/bids seemed to be adequate?

The alternative to the acquisition was to prolong the existing contract of the old mainframe system, until further replacement was arranged.

Was planning performed as thoroughly as needed?

No, we could have planned in more detail in the project, especially to help people responsible in the subprojects. We chose not to do this because the co-operation with the supplier starts first after we have selected a bid. Further planning was then performed.

When the supplier was selected, what plans did you update?

When the supplier was chosen we had to decide on a project process that was going to be used for interaction and communication. This involved the major part of the planning. One might say that we delegated some of the planning decisions onto meetings further ahead. This meant that the supplier could be involved to a higher degree in decisions that we needed help with.

What would have changed the project if the plan had been more accurate?

Not much, the initial plan did not reach a long way ahead in time and was quickly a thing of the past. The updates of it was not conducted on regular basis.

How was the organization for the project formed?

The project has had one full-time employee, the other 40 persons involved in the project has worked with the project in addition to the ordinary tasks at work.

The project involved many departments of the city and representatives from each department were involved in the project. Each department had a responsible person. The end users of the acquired software worked in the departments.

Each department in the city has a technical person for the administration of technical information system issues.

The method of working, also decided on in planning, how did it work?

We chose a model for working closely with the supplier that is called extreme programming[1]. End-users work in tight co-operation with suppliers when evaluating the product. At meetings new functionality is elicited in co-operation with supplier and end-users.

Did you have any plans on sanctions if the supplier was contracted but nothing happened from the supplier's side?

The sanctions are specified in the in the acquisition foundation in the form of penalties. The deal with the supplier is arranged as part payments. The complete amount for the system was divided into two equal parts. The first part was divided into four part payments. These payments were fixed at part deliveries of the system.

How were requirements elicited?

Representatives from the departments affected by the acquisition were present at the initial brainstorming meeting. The scenarios we came up with were business oriented. They displayed tasks that we perform today with the current system, however not connected to the old system.

We did not want to overlook any of the possibilities that such a program could provide so we made a survey of similar programs on the market. This resulted in a more stringent requirements specification.

Our strategy was to have close co-operation with the supplier and having the users to evaluate the program as often as new functionality had been added.

Do you consider the requirements to have covered the desired functionality of the system?

At first, the basic functionality in the system corresponded very well with the requirements specification. This was evaluated at each department involved in the project. But since nothing was defined concerning the user interface in the require-

1. This refers to the development method described in Chapter 5.4

ments specification, the departments had diverse opinions on the functionality of the user interface.

We also noticed in an early stage of the project that the interfaces towards other systems were insufficiently specified. We had left this part entirely out from the requirements.

Another thing that we left out was the description on how data should be processed in the program. The data is for example used for economic calculations. The output of the program, when data was processed, should be passed on to the external project "Ny ekonomimodul". Nothing of this was specified.

Did the requirements specification function as a common basis for communication when users met with developers?

Responsibility has been delegated from the project management to the various departments, where developers meet with local users. The synchronization between departments has not functioned in a satisfactory manner. The additions to the requirements specification that meetings with developers generated, has been engrossed in meeting minutes. The minutes should be available in electronic form for all personnel involved in the project. However the protocols was not given the priority that they needed. Thus not functioning as reliable communication between departments. Hence the departments have not had a fixed configuration of the system to proceed from. What the basic system functionality comprised was a rather floating concept.

Many of the users in the departments have been badly prepared at meetings with developers. The users have not studied up on the scenarios, the requirements specification and the minutes. The effect of this was that developers and users waisted much time on deciding on things that already were decided on. Users ignored the requirements when it did not favour their decisions on functionality.

Since various opinions of the basic functionality and the user interface existed, do you think that this could have been straightened out by having a clear goal or requirements reflecting the goal already established before the meetings?

Disagreement has been for example on what basic screens should look like, for example address-information and how to handle these. This could have eliminated by specifying this better or by gathering the personnel and show prototypes until agreement was met. We did not have any routines for solving this kind of disagreement.

Was the requirements specification updated as the requirements changed? For example when the users conducted tests with the developers in parallel in the various departments, was the requirements specification updated to reflect these changes?

As mentioned earlier an electronic file sharing system supported this. The changes were documented as protocols from meetings involving developers and end-users. The changes were documented but with a delay. To counteract the delay we sat a deadline for the minutes to the next day. This rule was seldom followed.

Why?

Except for writing the minutes, users had their ordinary tasks to perform. Of cause they where compensated for the project involvement, but their normal amount of work could not be reduced.

How was the evaluation of the system conducted?

End-users in the departments received a beta program to perform the test on. The tests were documented in relation the requirements specification and the result of the program. We had routines so that each department could document the result in a uniform manner.

When we did the first test we discovered that the period for testing the program was too tight for the users. They did not manage to test all the functionality at the time fixed. The planning and instructions for the realization was not explicit enough. The protocols generated were of varying quality and to some extent imprecise and vague. This probably depended on that the instructions were indistinct.

In some departments, the persons testing the system had limited experience with Windows. These users never worked with this environment before. They needed extra time to learn about basic functionality of the graphical user interface. The had quite limited understanding of the possibilities with graphical user interfaces since they had only experience with command line interfaces.

Meeting with the suppliers' developers took place after the evaluations. In these meetings if often turned out that users took functionality for granted. This means that this functionality not was communicated to the supplier. The meetings could sometimes be somewhat of a brake-block since a lot of new requirements were generated. This lead to that the supplier was slow on fixing the problems with the current version of the program. Instead the supplier worked with new the functionality. The new requirements and the functionality taken for granted have lead to delays.

When we read some of the test records, it was clear that users tried to form the new system into the shape of the old. We thought that this was a pity since the old system probably not was the optimal solution for these information-handling routines.

How was the responsiveness from the supplier after the evaluations?

The number of remarks/faults in the test records tended to increase without earlier remarks being dealt with. The supplier did not report if remarks/faults where dealt with or not in new versions of the program. This made the users more confused when performing tests. We also thought that the time between a meeting and the possibility to see results was too long. It neither favored the time plan nor the quality.

Were decisions in meeting documented in a satisfactory manner?

Writing minutes while meetings are in progress has not worked really well. The issues that have been overlooked are: decisions, time for completion of a task and the responsible for activities has not been appointed. List of activities that was going to work as a programme was not created.

Which roles in respective organization were present at the evaluations?

At the evaluation of the program, the end-users and developers where present. At the meeting where the entire projects was followed up, the project management from both supplier and acquirer was present.

Were these the right representatives from the organization to be present?

Yes, we think so. The responsible persons in the departments were the ones that decided on who was going to be present at the meetings. It was defined in their role that they had the responsibility to see to that the right competence was present at meetings. In some cases the right representatives might not have been present to bring the most to the purpose of the meeting. This might be in the eye of the beholder since it was up to the responsible person in the department to decide on that.

Has the incremental approach of working been satisfactory?

The functionality that we elicited at meetings was not divided into increments, that were going to be delivered at a set time.

May I call it evaluation cycles ending with a meeting between developer and user? How were these cycles planned, and how did they work out?

We have talked about some of the benefits and some of the drawbacks earlier. Users are now using a system that is directly tinged by their close involvement with developers, this makes them motivated for using it. The cycles were not planned in detail. The functionality that was proposed for the next version was not scheduled, it was built in gradually.

When did it end?

It ended after the summer-brake. When everybody came back we felt that it was enough of new requirements. We had to start finishing off the system.

Were the costs of disposal of employees considered when making plans for costs?

No, it was not. In our case we concentrated on the cheap price given by the supplier for such a complex system. Even if users were involved to a high degree we felt that it was an expense that we could spare. Maybe if suppliers had been more expensive we would have considered cutting costs that involved users.

Were art of employees for disposal given any thought?

No. The persons involved most of the time was project management and end-users.

When the employees are on loan to the supplier for requirements elicitation, testing etc. are they relived from their ordinary task of duty?

It exists from the Malmö city point of view a number of critical time periods when the project members can not perform any work in the project since their ordinary tasks takes a higher priority.

The project members who work in the project are sanctioned. Their work in the ordinary organization remains and the project management has sensed that the project come in second hand several times. The lack of a project plan with a schedule makes it a lot harder for the project members to plan their ordinary work and the work in the project.

What were the actual costs of disposal of the personnel during the project?

The project had one full-time employee and about 40 persons working with the project and with other tasks as well. We do not have any figures of the time that they have disposed to the project.

Was the employees responsibility and authority established?

Yes, in an early stage in the project.

The project member's power of initiative has been weak. Seldom contact with the supplier has been made on their own initiative. The responsible persons in the departments have been badly prepared on meetings. Their part of the projects in the departments are not pushed enough. This may depend on the insecurity of what authorities and responsibilities they had.

In turn it is also hard to assert oneself against the corresponding part at the supplier. It may also be a reaction of the supplier mentality when meeting with the users. The mentality had been: "don't worry" and "laissez-faire". The project management had wished for more pressure on the departments from the supplier.

Were responsibilities for the introduction of the program established?

We had in good time chosen persons to be super-users or normal users. At an early stage users was introduced to the system. It gave the users a chance of understanding how the system was going to affect them. Many users showed that they had commitment to the project, probably because they understood what happened and that they got the chance of being involved from the start. When the system was launched it gained acceptance and users had motivation for using it.

A.2 Case study B

A.2.1 Description

In 1998 Lund kommun acquired a web site. The purpose was to reach one step further to the 24 hour availability of community services. The database and tools would give the give the personnel in the departments an easily managed tool for publishing information with a consistent layout on the web site.

The acquisition bid that was accepted to a cost at 386000 SEK as a one-time fee. The annual maintenance fee was specified to 72000 SEK. The feasability study before the acquisiton took about five month and the project took about 9 month to complete. The project budget was crossed and the project was 3 months late.

The interview below is with the project manager responsible for the acquisition. The interview was tape recorded to maintain the uniqueness of the dialog.

A.2.2 Interview characterization

This interview was more open than the one from Malmö displayed above. The questions were not asked in a particular order. The interviewee talked freely around the subject as follow-up questions were asked.

A.2.3 Interview

You acquired the web site that Lunds kommun use today in 98. What was the purpose of the acquisition?

We had a web site earlier, but with that website, people were working manually. There existed no comprehensive control over the old web site. There was no person that delegated responsibility. Persons in the departments added any material to the web site without checking with others if this was legitimate. In addition to this the departments were segregated. One of the sides embraced technology. This group that was interested in technology was also very competitive against other departments. They wanted to control the information flow. The other side was hindered by the technology or not interested. Imagine that ten of the departments are really interested, five are not very interested and five are not interested at all. In all departments exist people that do not have time, interest or adequate knowledge to be a part of this information flow to the public.

The technique that we acquired was supposed to work as basis to form uniformity in the departments. Everyone should be able to add information without extensive experience of some obscure computer tool. We wanted to acquire a database tool that all information was stored in. The layout of the information should be uniform without any individual contributions in the departments. The graphical form was supposed to be uniform as well as the navigation of the web site….eh, a throughout thought of way to publish information. We also wanted that people not should have to learn to publish information more than absolutely necessary. We did not like that people should have to go through a weeks training just to publish some documents.

Did you think that you covered all these aspects in the request for proposal?

Er, yes. (Paus). All of the suppliers that answered to the request for proposal had tools to provide us the desired functionality. We did our request for proposal in 1998 and it is 2001 now.

It might be hard to remember all the details.

In that time 98, it did not exist many tools ready for use to fulfil our needs.

It did not exist many tools of this kind in 98.

There were a few tools that suited us. It was hard to choose between the tools. Each tool had some features that were extra attractive.

Was it hard to choose between the different tools?

I think that there are two aspects when you perform an acquisition. You want to follow the right procedure for choosing the right supplier. This part is the part that controls the acquisition. In the meaning that you have to chose on the criteria that you have elicited and included in the request for proposal. The other factor is if you think that the tools that you acquire are good or bad tools. Among those who returned bids on our request for proposal, was in line with our requirements specification.

Was it hard chose from the criteria that you had elicited? Could you choose the bid that you wanted from the criteria? Those criteria may be hard to elicit and form in an understandable manner especially as it is technology that you acquire?

Now, afterwards if I could do the acquisition again… err, I think that in this kind of branch that technology is… err, It is seldom that you have a chance to test the tools that you acquire. There are many things to make out afterwards when you have tested a tool. We would have liked to create our requirements specification under supervision of a corporation. Hardly any company demonstrated prototypes of their products. We did not find any company that offered this kind of service, the companies are afraid to reveal secret information to other companies. In a situation where an agreement not is reached, the companies are very hard to get help from.

Not very helpful when the contract is not signed, huh?

No, exactly. They give the illusion that everything is going to be solved when the contract is signed.

I what to know about how you think the planning was performed. As you said before, if you could perform planning again, what would you think about extra much?

We could hardly have planned any better then we did. We used wish lists from the departments when we performed planning for the project. These wish lists formed our requirements specification. We had scoured the market for products

before we made the request for proposal and we had our old website that we had worked with for a long time.

Did you have a time plan also, except for the requirements?

Err, yes, we had an informal time plan. We wanted to keep this plan for our selves because these things are so sensitive. What I think is spooky is peoples expectations, experienced and prejudices. This is all in a psychological plan.

So you experienced a pressure from people?

Yes, this IT-thing in general, it is required to be fast and you should be able to cope with any technical problems that you not are prepared for at all. In larger organizations, as in our organization, with 20 departments and 2500 workstations. (pause)… It is the all these users expectations that resist all the time. It is not possible for all of these users to come with their own opinions and expectations.

Ok, so who took their party in the communication with the supplier?

We had two people from the IT department, one person and myself, who represented the users. We also had an additional reference group. In this group a person from each department was represented. This was the same people that made the list of whishes for the requirements specification. This list of what people wanted and hoped for, took almost half a year to collect. It then took three months to compile the requirements specification that should be included in the request for proposal. We did several company visits during this time. Then it took tree months to evaluate the bids and select the supplier.

The people who met with the supplier, how often did they meet?

You mean the reference group?

Yes, the reference group.

With the potential suppliers, only the four of us were in direct contact.

After you had made you selection how did you communicate with the supplier?

We promptly had a grand meeting with the supplier.

What was the purpose of these meetings, how often were they held, how did the co-operation with the supplier work out? Do you experience that you had good contact with the supplier?

Immediately when we had selected the supplier I phoned them and arranged a meeting the next day. We had to sign the treaties. At this meeting we made a first preliminary time plan.

You made a time plan together with the supplier?

Yes, of cause. We have contract for about fourteen days and then we have a grand meeting.

What does it mean to have a grand meeting? Who is represented from your side at the grand meetings?

With a grand meeting, I mean that the representatives from the departments also are present. It is important that they receive this kind of direct information.

(In chorus)

Who did you meet from the suppliers side?

Yes, that I felt. In this branch people switch jobs very often. That I think is very disturbing in the communication with the supplier.

Ok, so this has been disturbing in the communication with the supplier?

Yes, for a customer this might be disturbing. But this phenomenon is not unique for us, it applies to all companies. So, that is something that you have to live with, otherwise it worked very well…and when we had the time plan we wanted to publish the new website in half a year.

Why did you set half a year as the limit for the development of the web site?

Because… in our own experience we know that it is not very good to delayed thing over long periods of time. Our organization is so large so things may be delay for a very long time. Then it might be delayed for a whole year. During this time techniques used in the product might grow old. People in the organization may feel that nothing happens in the project. That is why we had the grand meeting so that people in the organization feel that something happens in the project. The initial meeting should have the effect of a take-off.

You contacted at the grand meetings. During these meetings and at the meetings how could you measure the performance of the project?

We had regular contact. As a project manager I had almost daily contact with the supplier's project manager via the phone. Behind the project manager stood a web team with 4 -5 responsible persons.

Ok, so the web team met with people from the departments?
No…

(A sketch of the organization is drawn)
Lena starts to explain that the most intensive communication is running in the management. The phone rings, I turn of the recorder.

The interviewee is back from a few phone calls and I ask again about the communication between the web team and the end-users. How the organization was built around the project. Did you ever considered the costs of the involvement of the end-users? They have a regular job to attend to as well as the project…
Haha, no, we did not. We considered the costs for me as a project management.

Why did you not consider the costs of user involvement?
Because these calculations are very hard to perform. These calculations are not very relevant without adding another dimensions to the project. To have some use of calculations of these type you have to calculate the cost savings after the web site is published. For example a person might get ten phone calls a day about questions that after the web site is published someone can use a web browser to view. As we did not want to calculate the time that people use sitting in the phone answering questions doing their regular job.

Why did you not considered to calculate this?
The quantities for these calculations are too complex to ever calculate and maybe even to interpret.

I have some additional questions that we came in touch with earlier. Did you have an alternate plan if a supplier was not adequate when you received the bids?
(Pause) We would have waited for maybe half a year or so and made a new acquisition.

Was the method of working with the supplier decided on in planning?
What do you mean?

I, mean if you had divided the project into phases with milestones that the members in the organization could follow?

Oh, yes, we had several milestones in the project. Often they were set in conjunction with the grand meetings.

Was it clear to you what these milestones were and what should have been done at each milestone?

Yes, myself I had good idea of what was going to happen and where we were in the project all the time. Maybe if a person came late in the project it would take a month or so to work their way into the project.

Changing personnel might not only be a problem for the supplier, but also for you? For example if you had quit, it would have been a great loss for Lund…?

Yes, but the supplier personnel are very different from ours. Those working for the supplier are young people trying to make careers and often get other more attractive bids from other companies. In our organization we have a high average age, people has loans on their houses and they do not plan to make careers, so that is not a problem for us.

Were these discrepancies a problem when the supplier met with your organization?

Both yes and no. I know all the people in the departments and I had prepared them for this. Since I met the supplier at first, I had to accept the fact that I could not use their language at all, with all the fancy IT expressions that they use. In the departments that were least interested in technology I had paid visits and made jokes about the situation. This cleared the air at meetings with the supplier. We had adjusted to be aware of this discrepancy and we had laughs after the meetings if anything was misunderstood.

I see, how were the evaluations of the product conducted, at the meetings that you mentioned above maybe?

What do you mean?

You did have an acceptance of the product at the end, but how did you evaluate during the project?

We had a checklist of features that the old website comprised that was going to be transferred to the new one. When this was done the project was going towards the end. The only problem was that the list grew all the time with functionality that we did not know existed but still we was forced to have. I had a hard time to keep that list to contain the most important functionality.

You optimized the project to contain less functionality to be able to complete in time?

Err, no, we had the list that was gong to be transferred from the old website to the new, this takes time.

But if you imagine that you would like to optimize for example functionality, cost or time, which was what I meant…

Well, we had a very hard schedule to keep. The employees that had to work over from the different departments often controlled if I was present at my office at that late hours. They bought hot dogs outside my window and often asked to see if I wanted one to see if I was at my office. They even used the intern mailing system to see if I was online when they were. I was glad that I could work late hours all the time when they were.

That have to be a tough role to wear, as a project manager I mean.

(Silence)

Ok, I have some more questions to wrap the interview up… so, for example to transfer the knowledge of this project to the next, how do you perform such a thing?

(Long pause)

Do you talk to each other?

Of cause…

Interrupted by interviewer by the question: Do you for example write down what has happened during the project to be able to communicate it amongst the project members?

Yes I always took minutes at meeting that I sent out in the organization.

Allright, do you think that the product when finished covered the functionality that you were looking for?

We still work on the list (2001) that developed during the project. Now for example we have interactive form on the web for public users to fill out. We are working against the goal to be a 24-hour government. I must say yes to answer you question, we were happy with the product when it was finished.

The end-users, did they immediately start to work with the product? Were they as satisfied as you?

(pause) Yes, I think so. Today (2001) all the users have realized the potential of the web site.

So, how did it work when you accepted the product?

We had a month to evaluate it. So, we had a period when the users used it as for real.

Did you not have any problems when it came to use?

Oh, yes, all the problems were reported to me. The supplier's manager and myself sat in close contact to investigate what were real problems and what were supposed to be in that way.

How did the supplier respond to problems that arise?

They were very forthcoming and issues were solved fast.

That was all for my part. I just want to congratulate to a successful acquisition!