Critical success factors in software projects - a framework under scrutiny

(**HS-IDA-EA-03-310)**

Timothy Little (a00timli@ida.his.se)

Institutionen för datavetenskap
Högskolan i Skövde, Box 408
S-54128 Skövde, SWEDEN

Examensarbete på det systemvetenskapliga programmet under vårterminen 2003.

Handledare: Lena Aggestam

Critical success factors in software projects – a framework under scrutiny

Submitted by Timothy Little to Högskolan Skövde as a dissertation for the degree of B.Sc., in the Department of Computer Science.

[2003-06-06]

I certify that all material in this dissertation which is not my own work has been identified and that no material is included for which a degree has previously been conferred on me.

Signed: _____

Critical success factors in software projects – a framework under scrutiny

Timothy Little (a00timli@ida.his.se)

Abstract

As a means of addressing the failure rate of information systems Aggestam (2001) proposes a framework which aims to guide organisations in the development of this type of software system. Software is a common concept today and can therefore be anticipated in contexts other than organisations. Examples of such contexts can be given as: embedded software, scientific software and personal computer software. The literature informs that 20% of these software projects are failures and 46% experience cost and schedule overruns. In an attempt to address this failure rate the aim of this report will be to investigate if the framework proposed by Aggestam (2001) can also be applied in this type of software project.

Through a comprehensive literature study success factors pertaining to software projects where an organisational information system has not been built have been identified. These factors have then provided the foundation for a deeper interview study. It has been shown that the framework displays promising potential for use in this type of software project. A stable groundwork has also been laid for continued research in this area.

Keywords: framework, information systems development, software engineering, software, success factor(s).

Table of contents

Supplements:

1 Introduction

This section is intended as an overview and has thus concentrated on the most central topics that have been raised in the forthcoming pages.

1.1 Background

Clegg, in Leibowitz, (1999) observe that 90% of all information technology (IT) projects fail to meet their goals, 80% are over budget and 40% are abandoned. Leibowitz (1999) refers to a survey carried out by the Standish Group (1995) who found that 31% of new information systems (ISs) are cancelled before completion and that 52.7% of the projects completed are 189% over budget. Avison and Shah (1997) inform that IS projects can be large and complex and risk of failure is considerable.

Aggestam (2001) proposes to address the failure rate of ISs with a framework that aims to guide organisations in what considerations should be made before the IS project begins. An IS is a type of software system but with many additional components. Avison and Shah (1997) explain that an IS consists of: hardware, software, people and data where the software component consists of: systems software, applications software and specific purpose software.

The term Software is widespread nowadays and can be anticipated in a variety of application areas other than organisations. Pressman (1997b) identifies five such types of software: real time software, engineering and scientific software, embedded software, personal computer (PC) software and artificial intelligence (AI) software. There are a multitude of additional software systems where these software types feasibly could be applied. Sage and Palmer (1990) suggest the following: telecommunications, command and control, automated manufacturing, electronic mail and office automation. Norris and Rigby (1992) give additional examples with: radiotherapy machines in hospitals, automatic plane landing systems, high speed train braking systems, process controllers in nuclear and chemical plants.

McConnell, 1996, in Procaccino, Verner, Overmyer & Darter (2002) informs that 20% of software projects are failures and 46% experience cost and schedule overruns. Hoffman, 1999, in Procaccino et al, (2002) ascertains that failure rates for software development projects are as high as 85%. These figures indicate that software projects also suffer alarming failure rates. Perhaps something can be done to rectify this.

1.2 Problem

Aggestam (2001) has constructed a framework which proposes to guide organisations in what considerations should be made before the IS project begins. The framework is founded on success factors within the areas of information systems development IS development and organisational development. From these success factors two superior categories have been identified: stakeholder and objective. The third superior category, boundary, is not founded on success factors but is considered by Aggestam (2001) to be prerequisite for the entire framework.

Given that the framework proposes to improve the success rate of ISs, can the same framework be used to improve the success rates of other types of software project? This reasoning has led to the following formulation for the problem to be solved:

Does the framework show potential for application in software projects where the purpose is not to build an organisational information system?

1.3 Method

In order to obtain a suitable base from which an answer to the formulated problem can be derived it is necessary to conduct a literature analysis. This analysis will serve to identify success factors in software projects where the purpose is not to build an organisational IS, thus providing a stable base from which a deeper study can be built. By conducting an interview based study a deeper discussion, based on the material gathered in the literature analysis, can be initiated.

Berndtsson, Hansson, Olsson & Lundell (2002) inform that aspects such as *validity* and *reliability* require careful consideration when choosing which methods to solve a problem. Berndtsson et al (2002) advise that validity is the relationship between what you intend to measure and what you actually measure. Reliability is the accuracy of the method. The author of this report is of the opinion that literature analyses and interviews are valid and reliable choices for the range of the problem area.

1.4 Result

By both explicitly and implicitly aligning material in this report to success factors and processes in the framework a connection has been established. It has been shown that all success factors, with the exception of one: *meet business objective*, in the categories stakeholder and objective can be substantiated. The inability to support this particular success factor is addressed by McDermid (1990) who maintains that the domain of the problem to be addressed constitutes a major difference between the building of organisational ISs and other software systems. Wateridge (1997) ascertains that the ISs ability to support the business function of the organisation constitutes a factor for success. The author of this report is of the opinion that it is unlikely that an embedded software system or a specific PC application will be required to support the business function in the same way, thus motivating this discrepancy.

The alignment to the categories stakeholder and objective is explicit, thus confirming the framework's potential for use in pure software reports. However, the majority of the success factors identified in this project have not been aligned in this way thereby challenging the strength of the framework's potential for use in this type of project. By implicitly aligning the majority of the remaining success factors to processes in the framework further potential for its use can be suggested.

The work in this project has certainly contributed with the first stages of expanding the framework into newer areas of application. The work carried out has *not*, however, been sufficient in order to state that the framework *can* be used in software projects where the purpose is not to build an organisational IS.

1.5 Conclusions

This project has initiated the groundwork for expanding the framework's scope of use to encompass non IS software projects. In order to be able to state that the framework *can* be used in this area work still needs to be done in order to more explicitly align the implicitly aligned success factors. In addition, a better understanding of the framework's inner workings needs to be acquired in the context of a non IS software project. Once a more explicit alignment has been established guidelines should be formed outlining how the framework could be applied in this type of project. In order to validate the framework's applicability in a non IS project a case study should be conducted.

1.6 Outline

The report will commence with a background to the problem area. The framework proposes to improve success rates of ISs and thus the area of ISD will be reviewed. Dorfman and Thayer (1997) explain that the development of software is generally referred to as software engineering (SE) and as such SE will also be represented. A correlation will be established between these two main areas and it will be shown that SE is a subset of ISD, thus initiating a focal point for expanding the framework into other areas of software development. A full presentation of the framework will serve to familiarise the reader with the structure and workings of this mechanism.

A problem will subsequently be formulated and methods will be selected in order to solve the problem. A full presentation of all the gathered material will form the foundation for an analysis and the final result. A critical reflection over the work conducted and recommendations for future research will conclude this report.

2 Background

This chapter aims to provide necessary background information with respect to the problem area and to provide a definition for important terms.

Many information system (IS) and software projects are deemed failures. A framework proposed by Aggestam (2001) aims to address the failure rate of ISs by guiding organisations in the considerations that need to be made before the IS project begins.

The building of an IS is governed by IS development (ISD) (Andersen, 1994). The building of software is governed by software engineering (SE) (Dorfman and Thayer, 1997). The disciplines of ISD and SE share both similarities and differences. McDermid (1990) ascertains that the domain of the problem to be addressed constitutes perhaps the greatest difference between these two disciplines.

A framework proposed by Aggestam (2001) aims to improve the success rate of ISs. Given that a multitude of software projects are also considered failures; can anything be done to rectify this?

The remainder of this chapter is presented as follows: in section 2.1 the areas of ISD and SE will be introduced and a link will be established between these two disciplines. As part of this discussion a presentation of types of software will serve both to account for the variety in this domain and provide one of the focal points for the rest of this report. A presentation of the development processes affiliated to these disciplines will follow in section 2.2 with the intention of highlighting similarities and differences. In section 2.3 a discussion concerning characteristics of successful software and a successful IS will be presented. This section will serve to provide a foundation for section 2.4 where a framework for managing success factors in ISD projects will be introduced. This framework is of particular importance to this project and provides a point of reference from which comparisons can be made and will be referred to throughout the course of this work. The chapter concludes with a summary of the chapters major themes.

2.1 Information Systems development vs software development

The purpose of this section is to provide an introduction to two chief areas which will dominate the following pages. It can be commented that this introductory section discusses topics that may not yet have been covered in depth. The reason for this is simple. By presenting a high level of abstraction and thereby establishing a context, a structured discussion concerning the topics raised in this introductory section can more effectively be managed. Throughout the course of the following discussion the concept of an IS will be addressed. It would therefore seem prudent to start the discussion with a definition of this concept.

The concept of an IS has been subject to a number of definitions. A recurring theme among these definitions is the concept of a computerised IS (see e.g. Avison and Fitzgerald, 1993; Wood-Harper, Antill & Avison, 1985; Avison & Shah, 1997). The components of a computerised IS are discussed by Avison and Shah (1997) as including:

- *Hardware*
- *Software*
- *Data*
- *People*

Andersen (1994) maintains that an IS transfers information between people. This transfer is accomplished by expressing information in the form of data. As is highlighted by Aggestam (2001) data can only become information when it has been interpreted by human beings. Andersen (1994) interprets an IS as a system for gathering, processing, storing, transferring, transmitting and presenting information. An addition to Andersen's interpretation is provided by Yeo (2002) who observes that an IS stores, processes and delivers information relevant to an organisation in a way that it is of use to those individuals who use it. For the purpose of this project an IS is regarded as being computerised and consisting of the components outlined by Avison and shah (1997). The functions of an IS are effectively outlined by Andersen (1994) and therefore a working definition for an IS is given by the author of this report as:

> An IS is a system consisting of hardware, software, data and people used for gathering, processing, storing, transferring, transmitting and presenting information in an organisational context.

From the above definition it has been established that an IS consists of*: hardware, software, people* and *data* where software according to Avison and Shah (1997) is the aspect of the IS which will correspond to the automated procedures within the IS. Andersen (1994) informs that the building of an IS is accomplished through the process of ISD, which is an extensive task demanding competence in a number of areas. Avison and Fitzgerald (1993) observe that when a computerised IS is designed a software engineering (SE) phase is usually incorporated into the ISD process.

It can be commented that the incorporation of a SE phase into the ISD process is conducive to the traditional picture of ISD, which according to Andersen (1994) assumes that an organisation develops its own IS from beginning to end. Andersen (1994) explains that organisations who do not want to develop their own IS, which is a costly and demanding process, can purchase a standard system which can then be modified. Andersen (1994) explains that these systems are manufactured with the purpose of being implemented in organisations. According to Andersen (1994) such systems are available in a multitude of forms. Where a standard system is implemented and not modified an SE phase is not incorporated into the ISD process.

Two principle disciplines have been touched upon which require further explanation. The discipline of ISD is concerned with the development of ISs (Andersen, 1994). According to Avison and Fitzgerald (1993) an SE phase is incorporated into the ISD process. In order to clarify these two disciplines a deeper presentation will now be given.

Information systems development

(Clegg et al in Leibowitz, 1999) observe that 90% of all information technology (IT) projects fail to meet their goals, 80% are over budget and 40% are abandoned. Leibowitz (1999) refers to a survey carried out by the Standish Group (1995) who found that 31% of new ISs are cancelled before completion and that 52.7% of the projects completed are 189% over budget.

The statistics outlined above paint a grim picture of the success rate of ISs. This is obviously not the intention of ISD. Indeed, as Andersen (1994) informs ISD is concerned with the building of ISs, which is an extensive task demanding competence in a number of areas. In addition to knowledge of methods, tools and descriptive techniques Andersen (1994) advises that understanding and knowledge about the development context and its environment is essential. Avison and Shah (1997) would agree, maintaining that the development of ISs is a complex and difficult task requiring that considerations are made with regard to: technical issues, organisational policies, organisational culture and the effect of ISD on individuals in the organisation.

Avison and Shah (1997) highlight that ISD is considerably more than a technical undertaking. Five factors are identified by Avison and Shah (1997) which must be accounted for when undertaking an ISD project:

- *Organisational structure*
- *Organisational culture*
- *Technology*
- *Tasks*
- *People*

Andersen (1994) informs that the ISD process starts off by planning the forthcoming IS. Four chief phases in the planning stage are identified by Andersen (1994) as: business analysis, information systems analysis, principle formulation of the technical solution, formulation of the automised technical solution. When considering the term technical solution Andersen (1994) differentiates between those activities which will be handled manually and those where automisation is intended.

The IS should support the organisation it is implemented in (see e.g. Andersen, 1994; Avison and Shah, 1997). According to Andersen (1994) by planning the forthcoming IS it is possible to chart what problems and possibilities the business stands up against.

Software engineering

The development of software is generally referred to as SE (see e.g. Dorfman and Thayer, 1997). Software intensive systems according to Andriole and Freeman (1993) are hallmarks of the modern world. King (1988) observes that the term software is a generic term for all types of software program. It can therefore be considered that software can be anticipated in a variety of different contexts and applications other than ISs. These systems according to Andriole and Freeman (1993) pervade in the public and private sectors. Norris and Rigby (1992) give examples of such systems as being: radiotherapy machines in hospitals, automatic plane landing systems, high speed train braking systems, process controllers in nuclear and chemical plants. Sage and Palmer (1990) observe that the widespread use of computers has resulted in a plethora of information technology (IT) products. Sage and Palmer (1990) continue, maintaining that IT products and services based on computers such as telecommunications, command and control, automated manufacturing, electronic mail and office automation are common terms today. Computer aided everything; according to Sage and Palmer (1990) may well be a common term tomorrow.

Jones (1990) explains that SE was derived from the fact that organisations were incapable of managing large software projects. This situation is referred to by Jones (1990) as the software crisis which was characterised by the late delivery of expensive, unsatisfactory and unmaintainable software systems. The software crisis is a well documented phenomenon in SE literature. According to Dorfman and Thayer (1997) the purpose of SE was to introduce an engineering discipline to software development. Dorfman and Thayer (1997) continue, explaining that the impetus behind this addition was the attempt to solve or reduce the problems of late deliveries, cost overruns, and failure to meet requirements.

McConnell, 1996, in Procaccino, Verner, Overmyer & Darter (2002) informs that 20% of software projects are failures and 46% experience cost and schedule overruns. Hoffman, 1999, in Procaccino et al, (2002) ascertains that failure rates for software development projects are as high as 85%.

As was the case with ISD, software projects are also plagued by failure. The discipline of SE aims to develop software in a structured, disciplined, economical and reliable manner. These aspects are captured in the following quotations:

SE is defined by Bauer as:

> *"the establishment and use of sound engineering principles in order to obtain economically* [sic] *software that is reliable and works efficiently on real machines*"(Bauer, in Pressman, 1997a, p.57).

A further definition is offered by Dorfman and Thayer who define software engineering as:

> *"The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software, that is, the application of engineering to software"* (Dorfman and Thayer, 1997, p.ix).

These quotations capture the essentials of SE but emphasise different aspects. (Bauer, in Pressman, 1997a, p.57) focuses on the economics and reliability of the product:

> *"…economically* [sic] *software that is reliable and works efficiently on real machines*".

Dorfman and Thayer (1997) emphasise the approach aspect choosing terms such as systematic, disciplined and quantifiable. In other words, the engineering aspect of SE is more heavily emphasised in the latter quotation. Due to the fact that these quotations highlight different, and indeed, very valid aspects of this discipline the author of this report feels that their inclusion in this report will serve to accurately portray the essence of SE.

Pressman (1997a) considers a layered approach to the discipline of SE. A high level of abstraction is captured in a process model (see figure 1). The bedrock for the process model according to Pressman (1997a) is the organisational commitment to quality. Pressman (1997a) maintains that total quality management fosters a continuous improvement culture which in turn fosters increasingly more mature approaches to SE. Pressman (1997a) advises that the process layer outlined in figure 1 defines a framework for the key stages which must be established for effective delivery of software. The process layer according to Pressman (1997a) is the layer that enables, by holding the layers methods and tools together, rational and timely development of computer software. Jones (1990) adds that a software development process model describes how these stages are linked together in order to be able to

trace the entire life history of the software product. With regard to the layers methods and tools Pressman (1997a) informs that methods provide the technical "how to's" for building the software and tools provide automated or semi automated support for the process and the methods.

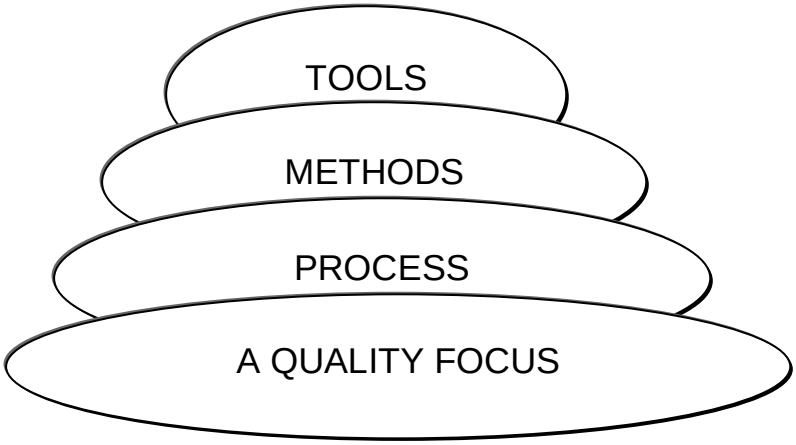

Figure 1: software engineering layers (adapted from Pressman, 1997a, p58).

The layers outlined in figure 1 serve to represent a high level of abstraction when considering SE. As has been highlighted by Pressman (1997a) a quality focus is the underlying concept that supports SE.

An introduction has been given to the areas of ISD and SE. It has been established that the development of software is generally referred to as SE (see e.g. Dorfman and Thayer, 1997). It has further been established that the development of a computerised IS inevitably involves an SE phase within the ISD process (see e.g. Avison and Fitzgerald, 1993).

According to McDermid (1990) there are fundamental differences which separate the development of an IS from other types of software based systems. The domain of the problem to be addressed, according to McDermid (1990), constitutes the greatest difference between the two domains. As an example, McDermid (1990) compares the development of an IS to the development of an embedded system. According to McDermid (1990) the parameters governing the embedded system could easily be established, whereas establishing what information should be handled by the IS requires a multitude of decisions. McDermid (1990) continues, highlighting the issue of requirements and particularly the gathering of requirements as being highly dissimilar between the two systems. In the embedded problem the solution is dominated by physical, quantifiable factors often of a mathematical character. This is not the case with the IS. McDermid (1990) explains that the specification governing the IS must clearly state what information is required by whom, in what form and when. In addition, the information in the IS is usually shared amongst many users.

Figure 2 captures the essence of the discussion so far. The development of ISs is accomplished through ISD. According to the given definition an IS consists of hardware, software, people and data. The IS software, as suggested by Avison and Fitzgerald (1993), is developed in a specific SE phase.

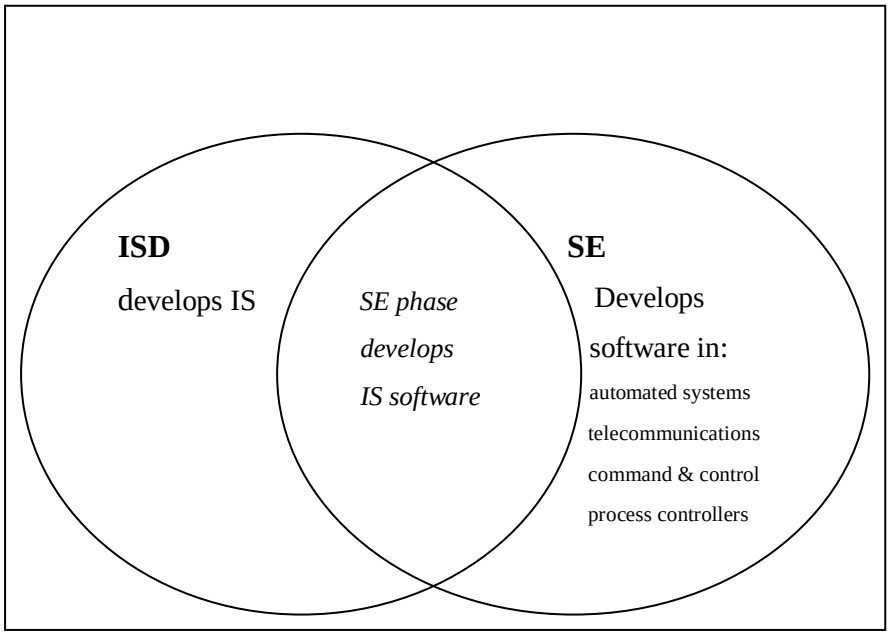

Figure 2: diagram linking the disciplines of ISD and SE.

The development of software is generally referred to as software engineering (SE) (Dorfman and Thayer, 1997). It has been suggested that software can be anticipated in a variety of contexts other than ISs. It has further been suggested that there exist software systems that are not ISs. Examples of such systems have been given by as: telecommunications, command and control, automated manufacturing, electronic mail and office automation, radiotherapy machines in hospitals, automatic plane landing systems, high speed train braking systems and process controllers in nuclear and chemical plants (see Sage and Palmer, 1990; Norris and Rigby, 1992). This has been incorporated into figure 2 where it is suggested that SE governs the development of software in such systems.

It should be commented that these systems are undeniably complex phenomena that inevitably are realised through complex processes. The particular processes' surrounding the development of such systems is not of interest to this project and consequently has not been discussed. What is of interest, however, is that these systems consist of software and as a consequence are also affected by SE.

Types of software

A discussion concerning software systems naturally lends itself towards a more specific discussion concerning the components of such a system, i.e. software. The discussion will now continue by presenting different types of software and their purpose.

The development of software both in ISD and in other contexts has been referred to as SE. King (1988) observes that software comes in three major varieties: *operating system software, firmware* and *applications software*. King (1988) explains that operating system software is responsible for managing mundane tasks such as scheduling and resource management. Firmware is a special type of operating system whose purpose is to customise the computer to a particular set of requirements. Applications software performs the manipulation and transformation of data. Fox

(1982) would agree on all points but one. Fox (1982) defines three overall software types as being: *systems software*, *applications software* and *support software*. Fox (1982) informs that support software enables programmers to create the software that runs at run-time. According to Fox (1982) compilers and assemblers belong to this category. In the context of an IS Avison and Shah (1997) inform that IS software usually comprises of:

- *Systems software*: Avison and Shah (1997) inform that systems software manages the operations and controls of the IS so that its use is optimised. This would be endorsed by Fox (1982) who adds that systems such as operating systems and database management systems (DBMSs) fall into this category of software
- *Applications software*: according to Avison and Shah (1997) applications software performs a specific set of tasks related to business functions. Examples of such tasks are given by Fox (1982) as being: payroll, inventory, message switching, and reservations.
- *Specific purpose software*: Avison and Shah (1997) explain that specific purpose software is specifically developed for a particular purpose within a specific organisation

It has already been established that software is not only restricted to ISs. Pressman (1997b) identifies the following software types:

- *real time software*: programs that monitor real world events as they occur
- *engineering and scientific software*: applications range from astronomy to volcano logy, from automotive stress analysis to space shuttle orbital dynamics, and from molecular biology to automated manufacturing
- *Embedded software*: embedded software resides in read only memory and is used to control products and systems for the consumer and industrial markets.
- *Personal computer(PC) software*: word processing, spreadsheets, computer graphics, multimedia are but a few of hundreds of applications
- *Artificial intelligence (AI) software*: AI software makes use of nonnumerical algorithms to solve complex problems that are not amenable to computation or straight forward analysis.

The software types outlined above represent the breadth of application areas and indicate that software can be anticipated in variety of contexts. The specific software components of an IS have been identified by Avison and Shah (1997) as: systems software, applications software and specific purpose software. Pressman (1997b) has identified five additional types of software that are not affiliated to ISs. This non-affiliation to ISs is of specific interest to this particular project and the software types identified by Pressman (1997b) thereby constitute an initial focal point for the remainder of this project. This topic will be readdressed in chapter 3.

It has earlier been established that fundamental differences arise when regarding the development of ISs compared to other software based systems. It would therefore seem prudent to closer examine the ISD and SE development processes.

2.2 The IS and SE development processes

A presentation of the ISD and SE development processes will now be given.

The ISD process

ISs can be developed with the help of a lifecycle (see e.g. Andersen, 1994; Avison and Fitzgerald, 1993; Avison and Shah, 1997). The lifecycle according to Avison and Shah (1997) was introduced to enable production of ISs in a more systematic, controlled manner by providing a useful framework within which to consider various tools and techniques.

Révay (1992) informs that a system's lifetime is given as that time which passes between the initial idea of a proposed system to that time where the system ceases to be. A key element of the lifecycle according to Avison and Shah (1997) is that the development of a system evolves through a logical, consistent process without omitting a phase. The ISD lifecycle can be considered a model in the life of an IS.

Figure 3 exemplifies what is known as the waterfall model, which is a well established development framework in both SE and ISD. The model uses the waterfall metaphor because it signifies the downward direction of the process, i.e. that the development progresses from one stage to the next. Figure 3 has been adapted from McDermid (1990) and represents a specific partitioning of the ISD lifecycle. In addition to the partitioning outlined in figure 3 McDermid (1990) advises that the waterfall model can be represented by the following stages:

- *feasibility study*
- *systems specification*
- *outline design*
- *detailed design*
- *system development*
- *testing*
- *implementation*
- *review* and *maintenance*

The latter partitioning would be in accordance with Avison and Shah (1997) who outline the following stages:

- *feasibility study*
- *system investigation*
- *systems analysis*
- *systems design*
- *implementation*
- *review and maintenance*

Figure 3 offers a high level of abstraction when considering the lifecycle concept and encapsulates the main stages of the IS development stages. As is highlighted by

Avison and Shah (1997) the overall approach and its constituent activities are broadly similar irrespective of what they are called.

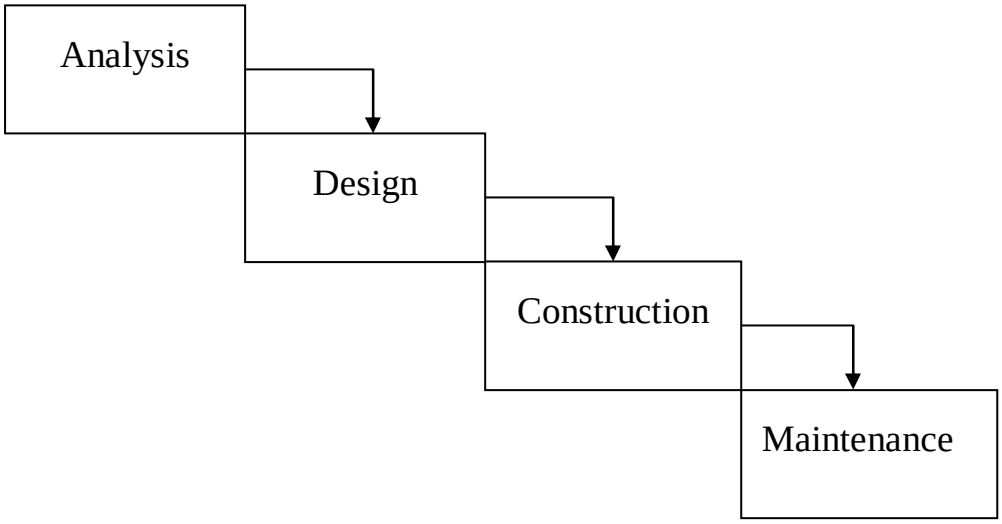

Figure 3: the waterfall model of the information systems development lifecycle (adapted from McDermid, 1990, p. 2).

It is apparent that the waterfall model can be partitioned in varying degrees of detail. In order to more clearly explain the stages of the lifecycle the partitioning offered by Avison and Shah (1997) will be used as it breaks the analysis stage of the lifecycle into three distinct activities: *feasibility study*, *systems investigation* and *systems analysis*. An explanation for the contents of the phases in this particular partitioning will now be given:

Feasibility study: during this phase the systems analyst(s) identify the initial framework for the application.

Systems investigation: Once approved the system must be planned in more detail. This will inevitably include identification of problem areas, data types, establishing the system's boundary, identifying processes that are performed on the data etc.

Systems analysis: analysis of the system is carried out in detail to determine the requirements of the system. This phase should result in a detailed description (logical model) of the system required by the clients

Systems design: this phase is concerned with designing a system that is compatible with the identified requirements. Hardware and software alternatives are considered. The logical model is translated into a physical design.

Implementation: during this stage the IS is built. This subsequently involves coding the programs, designing the operation procedures and producing relevant documentation.

Review and maintenance: this stage encompasses those problems encountered when using the operational system.

The SE development process

The concept of a lifecycle was introduced into SE by Royce as early as 1970 (Comer, 1997). The original software lifecycle model constituted what has been called the waterfall model (Comer, 1997).

Sage and Palmer (1990) maintain that software development lifecycles have their roots in engineering methods and more specifically in the processes and procedures of systems engineering. Sage and Palmer (1990) state that the use of the systems engineering approach to systems development and management is intended to result in a tailorable software development lifecycle that addresses the specific software development needs from both technological and management perspectives.

Jones (1990) observes that software has a lifecycle just like any other commercial product. Sage and Palmer (1990) maintain that the use of the conventional waterfall model has demonstrated that better software will result from the careful and systematic use of a software development lifecycle. Jones (1990) explains that the waterfall model anticipates that software moves in an orderly manner through each stage of its life. Jones (1990) continues, stating that the waterfall model provides a particularly strong framework within which to develop large software projects.

Figure 4 exemplifies the original waterfall model for software development which is generally attributed to Royce (see e.g. Comer, 1997). As was the case with the waterfall model in ISD, the waterfall model in SE can be partitioned in varying degrees of detail. Boehm, in Sage and Palmer (1990) refers to the following partitioning:

- *Systems requirements*
- *Software requirements*
- *Preliminary design*
- *Code and debug*
- *Test and preoperation*
- *Operations and maintenance*

Pressman (1997a) captures the essential features of the SE lifecycle in the linear sequential model which recognises the stages of the lifecycle as: *analysis, design, code* and *test*. The maintenance phase would appear to be missing from the latter partitioning and therefore the author of this report feels that Pressman's linear sequential model does not accurately represent the software development process.

An explanation for the stages of the SE lifecycle will now follow. The partition referred to in this explanation is that which has been identified by Boehm, in Sage and Palmer (1990). This partitioning is essentially identical to that outlined in figure 4 with the exception that the partitioning outlined by Boehm, in Sage and Palmer (1990) is felt by the author of this report to more explicitly highlight the nature of the activities associated with each stage. As an example, consider the stage *implementation* identified in figure 4. The corresponding stage identified by Boehm, in Sage and Palmer (1990) is referred to as *code and debug* which arguably is a more complete description of the activities associated with that particular phase.

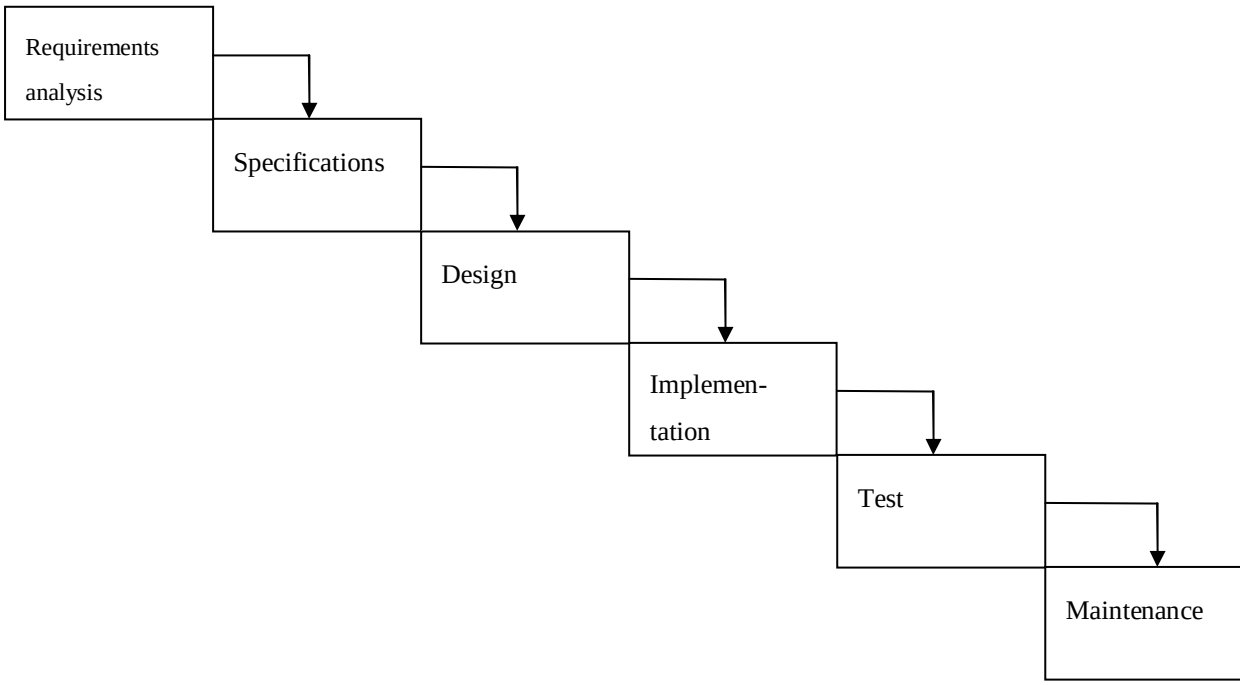

Figure 4: waterfall model of software development (adapted from Sage and Palmer, 1990, p.51).

An explanation for the contents of the phases of the chosen partition will now be given:

Systems requirements: in this phase it is assumed that the systems analyst is sufficiently informed as to the intended function of the system so as to be able to develop system level requirements to an acceptable level of detail from which a preliminary design can be initiated.

Software requirements: this phase is concerned with the software to be developed, the data and information that will be required, the performance and the various interfaces. This phase should result in a software requirements definition.

Preliminary design: a preliminary software product design is constructed in this phase. The chief purpose of this design is to enable further interpretation of the software requirements. This phase should result in a micro-level definition of the data structure, software architecture and the procedural activities that must be carried out in the next phase.

Detailed design: this phase is concerned with the definition of the program modules that are necessary in the preparation for the writing of code.

Code and debug: the detailed design is translated into machine-readable form. The resulting code is compiled and executed. When "bugs" are discovered debugging and recoding takes place to validate the integrity of the code.

Testing and preoperation: the individual units or programs are integrated and tested as a complete system to ensure that the system's requirements have been met. After testing the software is operated under controlled conditions to verify and validate the entire package.

Operations and maintenance: this stage encompasses those problems encountered when using the operational system.

The differences and similarities between ISD and SE

An account for the development processes in ISD and SE has been provided. The lifecycle model used to describe this process is the waterfall model which is a well documented phenomenon in both ISD and SE related literature. The waterfall model is perhaps one of the most widely used models in ISD and SE and as a consequence is felt by the author of this report to provide a suitable object for analysis. Indeed, as is highlighted by Boehm:

> *"The waterfall model has become the basis for most software acquisition standards."*(Boehm, 1988, p.417)

When comparing the development processes of ISD and SE it becomes apparent that certain differences are present. Firstly, it can be commented that the scope of both development processes are vastly different. In the ISD lifecycle three distinct analysis activities have been identified: feasibility study, systems investigation and systems analysis. It is the opinion of the author of this report that these three phases can broadly be referred to as the *"investigative"* stage of ISD. Indeed, as is highlighted by Avison and Fitzgerald (1993) the feasibility study aims to provide management with a list of solutions each of which have been described with regard to the human, technical and economic costs of developing an IS. As has already been highlighted by Avison and Shah (1997) the systems investigation phase involves planning the approved system in more detail and the systems analysis phase is concerned with establishing the approved systems requirements.

The SE development process would appear to ignore the first two activities of the analysis phase in ISD and instead begins with the establishing of requirements. This would be corroborated by Norris and Rigby (1992) who, when referring to the waterfall model, ascertain that the first phase begins with the definition of requirements. The establishment of requirements subsequently leads to the development of the software product which is accomplished by following the remaining stages in the lifecycle (see figure 4).

It has been established that the domain of the problem to be addressed constitutes a fundamental difference between the areas of ISD and SE (see McDermid, 1990). It has further been established that the ISD process incorporates an SE phase when a computerised IS is requested (see Avison and Fitzgerald, 1993). It is therefore possible to state that the development of the IS software is accomplished by following the stages outlined in the SE lifecycle (figure 4). It has further been suggested that software can be anticipated in a variety of contexts other than ISs. Suggestions for systems that are not ISs have already been offered. It would seem appropriate therefore to introduce into the discussion a presentation of the concept of software.

2.3 Characteristics of a successful IS and successful software

From the discussion so far it has been established that there are fundamental differences when developing an IS as opposed to other software systems. The domain of the problem to be addressed has been highlighted as being of significant importance with regard to this (see e.g. McDermid, 1990). Through an examination of

the ISD and SE development processes it has been established that both disciplines identify the waterfall model as being the most widely used development model (see e.g. Boehm, 1988; McDermid, 1990). The recognition of the waterfall model signifies an area of commonality between the two disciplines. This is of particular interest as this indicates the acceptance of a particular approach to development, i.e. a sequential, stage by stage approach.

It is the opinion of the author of this project that the production of successful software or a successful IS is the goal of the development process. This is arguably not the sole intention but perhaps the most important. The discussion so far has concentrated on specifics surrounding the development of software or ISs. In order to complete the discussion a presentation of those factors which are associated with success in software and IS projects will be given.

A successful software project according to Dorfman and Thayer (1997) meets its functional and quality requirements and is delivered within schedule and budget. Procaccino et al (2002) would agree claiming that in addition to time and budget successful software should also meet business objectives. Procaccino et al (2002) suggest additional success factors as being: *the degree to which the project achieved its goals, user satisfaction, effective project teamwork and the extent to which the software is used*. Johnson (2001) refers to *minimisation* as being a key success factor in software projects. Johnson (2001) explains that minimisation implies a project characterised by few project members and a limited duration. Reel (1999) suggests that the following factors characterise a successful software project: *the right team, low attrition, procedures for establishing quality, effective management, monitoring of progress, smart decisions and a process for learning from past mistakes.*

Avison and Shah (1997) observe that a successful IS benefits the organisation by enabling: *efficient operations, effective management and offering a competitive advantage*. (Flowers, 1996, in Yeo, 2002) defines an IS as a failure if any of the following occurs: *the system does not operate as expected, on implementation it is rejected by users and thereby under-utilised, the cost of the IS exceeds any benefits the IS may bring in its active life, the IS project is abandoned*. It can be commented that the reverse of these situations obviously implies IS success. Wateridge (1998) informs that deliverance on time, within budget and meeting requirements is a common mnemonic for IS success. According to Wateridge (1998) the criteria for success is much wider and that the aforementioned mnemonic essentially represents the contractors' viewpoint. By incorporating and considering other stakeholders views into the criteria for IS success Wateridge (1998) offers the following criteria for a successful IS: *it is profitable for the sponsor/owner and contractor, it achieves its business purpose in three ways (strategically, tactically and operationally), it meets its defined objectives, it meets quality thresholds, it is produced to specification, within budget and on time, all parties (users, sponsors, the project team) are happy during the project and with the outcome of the project.* Leibowitz (1999) considers a "lessons learned "repository for improving the likelihood of IS-project success. According to Leibowitz (1999)the following should be carried out for each failed project: *analyse how and why the project failed, cite causes/reasons for project failure, distribute these lessons learned to senior management, project management and members, create new guidelines on system development practices and procedures (for use in future projects).*

In summary it can be commented that the variables time, budget and requirements are crucial to the subsequent success of both ISs and other software systems. In addition

to these underlying variables it has been established that other factors play a considerable role when discussing success in such systems. As is highlighted by Wateridge (1998) the viewpoints of all stakeholders should be taken into consideration when evaluating success in IS projects. Similar considerations would appear to be addressed by Procaccino et al (2002) who consider variables such as user satisfaction and effective project teamwork when discussing success in software projects. A further point of comparison is the concept of learning from past mistakes. In the context of non IS software projects this aspect was addressed by Reel (1999). In the context of ISD the same aspect was referred to by Leibowitz (1999).

These success factors ultimately serve to enable the construction of software systems in as "trouble free" a way as possible. The ISD related success factors have provided a focal point for the establishment of a framework which proposes to address the issue of IS failure. It would therefore seem suitable to introduce this framework into the discussion.

2.4 Presentation of a framework for managing success factors in ISD projects

A framework for managing success factors in IS projects, as proposed by Aggestam (2001) is central to this project and a comprehensive overview will therefore be given. An introduction to the framework will be followed by a more in depth presentation of the identified success factors and their subsequent groupings into three categories (see table 1). A description of the framework and its components will conclude this subsection.

Aggestam (2001) proposes a framework for guiding organisations in which considerations should be made before the ISD process begins. The ultimate aim of the framework is to enable organisations to develop a successful IS. The concept of a successful IS has been well documented in various literary sources and has a strong affiliation to specifically identified success factors, something which has been addressed in section 2.3. Aggestam (2001) reasons that the preparatory phases of ISD are somewhat neglected, something which intends to be rectified by the proposed framework. Aggestam (2001) explains that by informing project leaders of success factors before the ISD process begins enables effective and efficient project adjustment.

Aggestam (2001) maintains that developing an IS ultimately results in organisational change:

"An IS is part of the organisation so to develop the IS is to change the whole organisation."(Aggestam, 2001, p. 1)

As a consequence of this the concept of organisational change is considered by Aggestam (2001) to be a sub-set of the ISD process.

The framework has been realised by identifying success factors within the ISD process. More specifically it can be commented that a particular emphasis has been placed on the areas of requirements elicitation and organisational development. Aggestam (2001) informs that the process of requirements elicitation is of particular importance within the process of ISD and consequently should provide valuable success factors. Aggestam (2001) suggests that development of an IS results in

organisational change. It follows therefore that success factors within the area of organisational development should also be considered.

A careful analysis of the literature regarding success factors in requirements elicitation and organisational development has enabled grouping of the identified success factors into two superior categories: *objective* and *stakeholder*. Aggestam (2001) conveniently refers to the objective as the destination of the journey, where the journey is a metaphor for the process of change in the organisation. The stakeholders according to Aggestam (2001) are particularly central phenomena that are both influenced by and influence all the other success factors. Furthermore, Aggestam (2001) has ascertained that identification of the proposed systems boundary is a prerequisite for the entire framework as everything else hangs in the balance of this careful definition. Aggestam (2001) has not identified specific success factors related to the boundary of a system but merely acknowledges its specific meaning in the context of the framework. It can therefore be concluded that the framework is founded on three superior groupings: *boundary, objective* and *stakeholder*.

For the purpose of this report the three superior groups: boundary, objective and stakeholder are of particular interest and will be referred to in the remainder of this report as the three cornerstones.

2.4.1 Success factors in the framework

Table 1 outlines the important success factors identified by Aggestam (2001).The identification of these factors is a prerequisite for the development of the framework and for the purpose of clarity has therefore been included. Identifying the systems boundary and its subsystems is a prerequisite for all other factors and has therefore not been categorised. Aggestam (2001) has filed those important success factors which have lent themselves to grouping under the headings *stakeholder* or *objective* accordingly.

Stakeholders • To achieve high user satisfaction • To involve the right stakeholder and manage their confidence and knowledge needs • To create a positive attitude • Championship, committed sponsor
Objective • Meet business objective • To define the objective • Accepted objectives • Good selection and justification
Remaining • To learn from failed projects • The organisational culture affects the process. Match the IS with organisational culture • The user always wants more • To know when enough has been discovered

Table 1: an overview of important identified success factors (adapted from Aggestam, 2001, p. 61)

Those success factors which have been filed under stakeholder are those factors which aim to involve the right stakeholders in the development process, generate a positive attitude and take care of the stakeholders needs. Although all the identified success factors aim to achieve this Aggestam (2001) informs that the success factor: *Championship, committed sponsor* has a slightly different focus. According to Aggestam (2001) this success factor concerns the whole project in the sense that the fulfilment of this factor is decisive for the commencement of a project.

For the purpose of clarity it should be mentioned that the attributative factors given in italics: *effective communication, human cognitive constraints, training and support* located under the category stakeholder are associated with the following success factor:

- To involve the right stakeholder and manage their confidence and knowledge needs

These attributative factors are given by Aggestam (2001) as a means of fulfilling this particular success factor.

The factors which have been filed under *objective* are those success factors which concern an objective that is being well defined, meets business objectives and is accepted among the stakeholders (Aggestam, 2001). These important success factors have been obtained through a comprehensive literature study within the areas of organisational development and ISD. Those important success factors which have not lent themselves to categorisation under the headings stakeholder and objective have been filed under the heading remaining.

The factors that have been filed under *stakeholder* and *objective* are identified by Aggestam (2001) as being critical and therefore important to the framework. It follows therefore that the factors filed under remaining should also be considered before the framework is constructed. Aggestam (2001) has shown that the factors filed under remaining are not relevant as they concern themselves with aspects outside the scope of the framework. It is, however, necessary to ascertain what these factors are concerned with:

- *To learn from failed projects*: Aggestam (2001) informs that the framework does not propose to enlighten organisations with regard to this aspect it merely requires that organisations actually do this.
- *To carefully document and analyse*: Aggestam (2001) informs that this factor affects the whole process and is not specifically concerned with the preparatory phases of ISD. Aggestam (2001) informs that this factor has a close affiliation to the category objective in the sense that a thorough definition of the objective is not possible without careful documentation and analysis
- *To match the IS with organisational culture*: according to Aggestam (2001) this factor is also concerned with phases outside the scope of the preparatory phases of ISD.
- *The user always wants more and to know when enough has been discovered*: Aggestam (2001) advises that these factors should be addressed in the development phase of ISD. Furthermore, the only considerations an organisation needs to make concerning this factor are incorporated into the category stakeholder

Consequently, Aggestam (2001) suggests a framework (figure 5) which aims to guide organisations in what considerations should be made before the IS project begins in order for the project to be successful. The framework has been realised by identifying important success factors (see table 1) within the areas of organisational development and the ISD process. The framework implies that upon identification of the system's boundary the organisation should analyse and describe the objective to be developed. Motivation of the stakeholders should then follow.

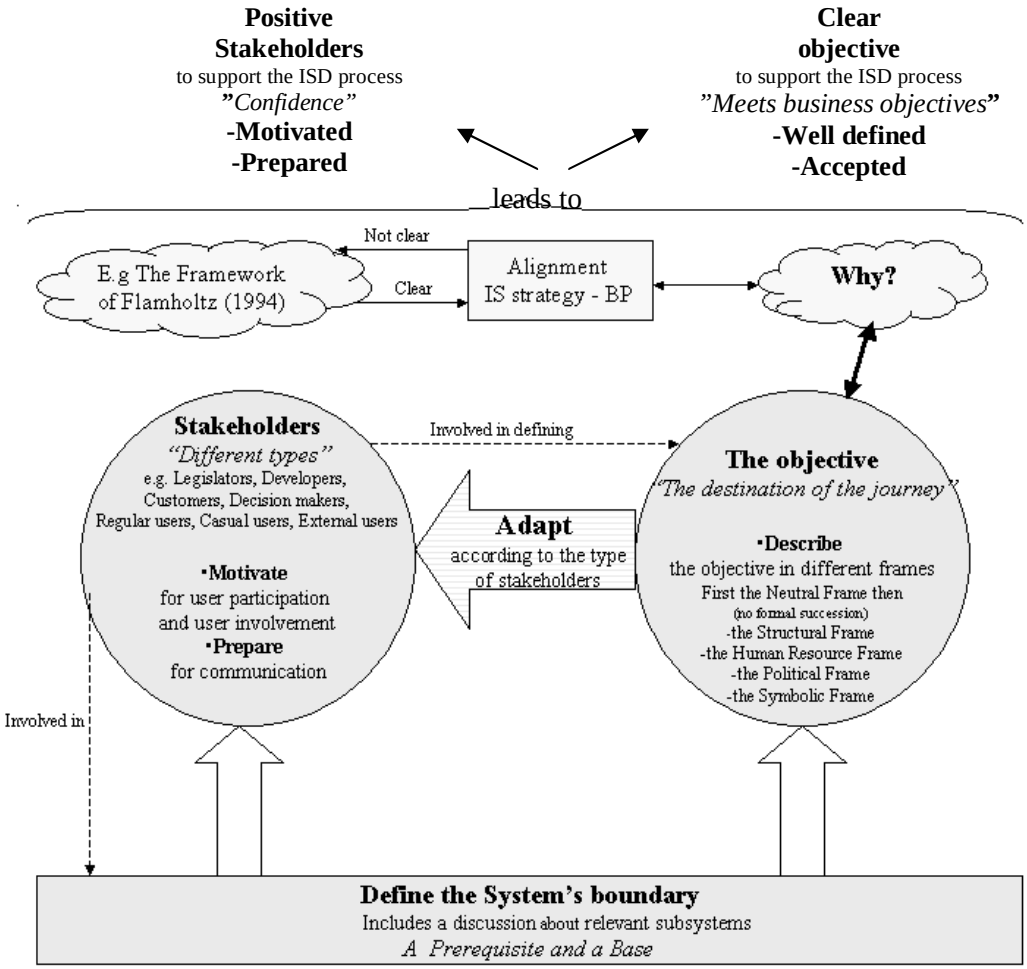

Figure 5: a framework for managing success factors in ISD projects (adapted from Aggestam, 2001, p. 64)

The framework implies that establishing the systems boundary and its subsystems is the first requirement for an organisation wishing to undertake an ISD-project. Once this has been achieved it is possible to discuss and define the objective and to identify relevant stakeholders.

The *motivation process* should focus on the stakeholders' knowledge and needs. The *motivation process* aims to describe the objective in such a way that individual stakeholders understand why the project should be carried out and, more importantly, how the project will affect him/her. The *preparation process* aims to improve the communication between stakeholders by familiarising the stakeholders with the terms and concepts to be used within the project.

Aggestam (2001) explains that the *motivation* and *preparation processes* aim to promote confidence and knowledge in the stakeholders about the prospect of change. Aggestam (2001) maintains that this is a prerequisite for stakeholder involvement and participation.

Aggestam (2001) informs that the objective should be well defined and described in different frames:

- *the structural frame:* in this frame the structure of the organisation is viewed
- *the human resource frame:* in this frame the relationship between people and the organisation is viewed
- *the political frame:* this frame views the importance of handling conflicts, disputes, power and decisions in organisations
- *The symbolic frame: this frame views symbols as embodying and expressing an organisations culture.*
- *The neutral frame:* this frame views the neutral perspective of the organisation. It focuses on aspects such as business mission, plan and size.

In each frame the following three levels of abstraction are considered:

- **What:** what do we need
- **Why:** why do we need it
- **How:** how are we going to implement it

Aggestam (2001) stresses the importance of the 'why' abstraction (see figure 5) claiming that an IS-project's objective must support the organisation if the outcome is to be a successful IS. Consequently, the objective must in every frame support the business objectives and the business mission which in turn constitutes alignment between the business plan and the IS-strategy. If this is not the case Aggestam (2001) suggests that a framework suggested by Framholtz may be applied to rectify this. As can be seen from figure 5, the category objective is connected to the category stakeholder by an arrow. This arrow represents an adaption process where the objective is formulated in such a way as to accommodate the needs of the various stakeholders. This is necessary in order to generate acceptance of the objective by the stakeholders.

By following the steps proposed in the framework it is anticipated that positive, motivated stakeholders and a clear, well defined, accepted objective can be realised.

In order to test its validity the framework has been implemented in an ISD related case study in a Swedish municipality. The resulting consensus (see Aggestam, 2001) was that the framework could be used effectively and therefore has potential for use in ISD projects. Throughout the course of this discussion the domains of ISD and SE have been regularly referred to. Indeed, it has been established that the ISD process incorporates an SE phase when a computerised IS is requested (see Avison and Fitzgerald, 1993). It has further been established that there exist software systems that are not ISs (see e.g. Sage and Palmer, 1990; Norris and Rigby, 1992) and that there exists software types which are not part of ISs (see Pressman, 1997b).

An interesting point is thereby raised. If the framework proposed by Aggestam (2001) is applicable in ISD projects where an SE phase is incorporated, is the same framework consequently applicable in other software projects that are not focused on an IS? It is the opinion of the author of this report that this suggestion provides a very interesting point for investigation.

2.5 Summary

The framework outlined in the previous section has been constructed with the purpose of guiding organisations in what considerations should be made before the IS project begins. The framework is an important addition to the discussion as a central point of

reference has now been established which will be referred to throughout the course of this work.

The framework aims to guide organisations in IS projects and therefore has a direct affiliation to ISD. The discipline of ISD has been referred to in this report as the development process concerned with the building of ISs (see e.g. Andersen, 1994). According to a working definition of an IS given for the purpose of this report in section 2.1 a computerised IS consists of: hardware, software, people and data. The development of a computerised IS inevitably involves an SE phase (see e.g. Avison and Fitzgerald, 1993; Ziya Aktas, 1987). SE has been referred to as the process concerned with the development of software (see e.g. Dorfman and Thayer, 1997). It has been established that there exists fundamental differences when comparing the development of ISs to other software based systems. McDermid (1990) advises that the domain of the problem to be addressed constitutes perhaps the greatest difference between the development of ISs and other software based systems.

In section 2.1 different types of software were presented. It was established that an IS consists of: systems software, applications software and specific purpose software (see e.g. Avison and Shah, 1997). Five additional software types were identified by Pressman (1997a) as being: real time software, scientific and engineering software, embedded software, PC software and AI software. It was further established that these five additional software types constituted a suitable delimitation for the purpose of this report.

By examining the development processes in ISD and SE an understanding for the development stages and development criteria surrounding these processes has been acquired. An area of commonality has been established in the waterfall model which has subsequently been reviewed from the perspective of both ISD and SE. Through this analysis it has been established that the scope of the ISD and SE development processes varied considerably. A comprehensive “investigative” phase was identified in ISD consisting of the phases: feasibility study, systems investigation and systems analysis. The SE development process was initiated with the establishing of system requirements and consequently lacked the comprehensive “investigative” phase identified in ISD.

In section 2.3 the concept of a successful IS and successful software was discussed. It has already been established that the five software types identified by Pressman (1997b) constitute a focal point for this report. It has further been established that the framework discussed in section 2.4 constitutes an additional focal point. Given that the framework proposes to guide organisations in the production of a successful IS and that the SE development process aims to produce successful software a discussion concerning the concept of success with regard to these products seems highly relevant. The essential features of the discussion so far are captured in figure 6.

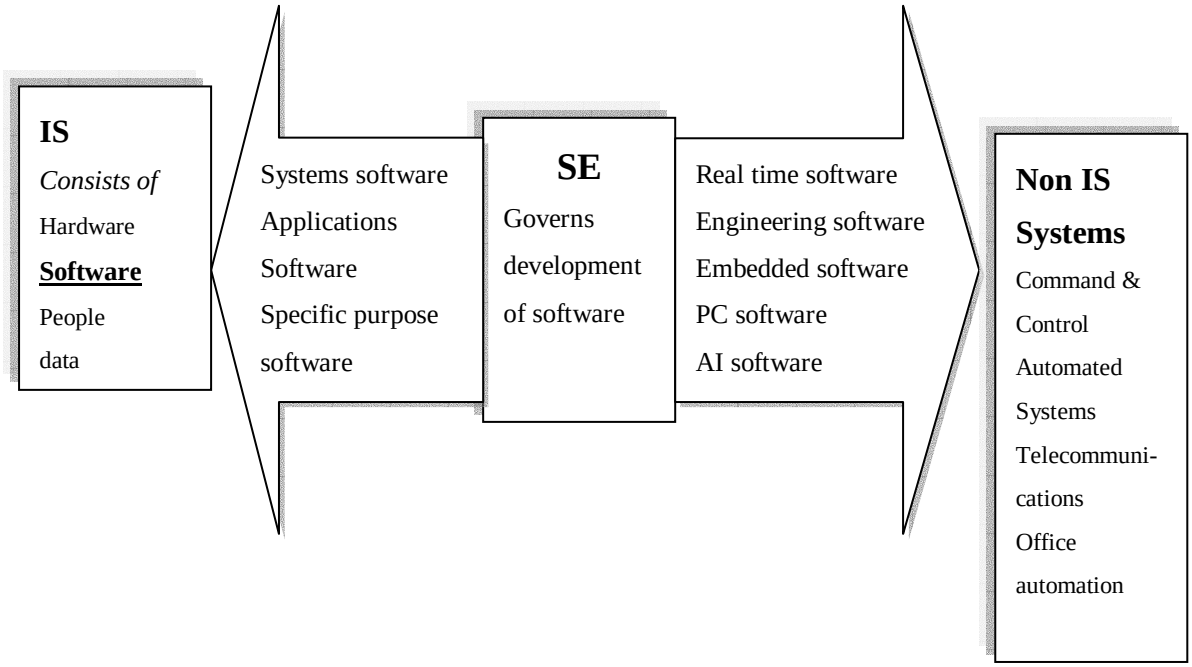

Figure 6: types of software and their affiliation to plausible systems

SE is the process which governs the development of software. The affiliation between an IS and its specific software types has already been made clear. In section 2.1 alternative software types were suggested which were not affiliated to ISs. These software types have been incorporated into the diagram. It should be made clear that with regard to the examples of non-IS systems the diagram merely suggests that the affiliated software types, shown in the corresponding arrow, may be associated with such systems. The diagram does not explicitly state that this is the case.

3 Problem

Aggestam (2001) suggests a framework (see section 2.4) which aims to guide organisations in what considerations should be made before the IS project begins in order for the project to be successful. A comprehensive presentation for this framework has been given and it has been established that three major groupings: boundary, objective and stakeholder constitute the foundation of the framework. These groupings have been referred to by the author of this report as the three cornerstones. In order to test its validity the framework has been implemented in an ISD related case study in a Swedish municipality. The resulting consensus (see Aggestam, 2001) was that the framework could be used effectively and therefore has potential for use in ISD related projects.

Throughout the course of the preceding chapters a multitude of topics have been addressed. The development processes of ISD and SE have been discussed and compared. It has been established that the ISD process inevitably incorporates an SE phase when a computerised IS is required (see e.g. Avison and Fitzgerald, 1993; Ziya Aktas, 1987). According to the definition of an IS given in section 2.1 an IS consists of hardware, software people and data. Avison and Shah (1997), when considering an IS, identify the software component as consisting of: systems software, applications software and specific purpose softwar*e*. It has been established that software is a phenomena that can be anticipated in a variety of contexts other than ISs. Andriole and Freeman (1993) refer to software intensive systems as a symbol of the modern world where such systems pervade in the public and private sectors. Sage and Palmer (1990) give examples of such systems as being: telecommunications, command and control, automated manufacturing, electronic mail and office automation. Norris and Rigby (1992) give additional examples with the following: radiotherapy machines in hospitals, automatic plane landing systems, high speed train braking systems and process controllers in nuclear and chemical plants. These systems are inevitably complex phenomena that consist of many components, amongst other things software. Pressman (1997b) identifies five software types that feasibly could be associated with such systems: real time software, scientific and engineering software, embedded software, PC software and AI software.

Given that the framework proposed by Aggestam (2001) aims to guide organisations in the development of a successful IS a strong affiliation to ISD is established. An affiliation to SE is also established due to the fact that the software component of the IS, systems software, applications software and specific purpose software, is developed in an SE phase incorporated into the ISD process. The development of software is generally achieved through SE (see Dorfman and Thayer, 1997). This would imply that the software types suggested by Pressman (1997b) are realisable through SE.

An interesting point is thereby raised. If the framework proposed by Aggestam (2001) is applicable in ISD projects where an SE phase is incorporated, is the same framework consequently applicable in other software projects that are not IS related and involve an SE phase? It is the opinion of the author of this report that this suggestion provides a very interesting point for investigation.

It has been established that the framework is applicable in ISD projects and therefore encompasses the development of the specific IS software identified by Avison and Shah (1997). Therefore, it would be interesting to see if the framework can be used to incorporate the software types identified by Pressman (1997b): real time software,

scientific and engineering software, embedded software, PC software and AI software. Figure 7 graphically portrays this intention.

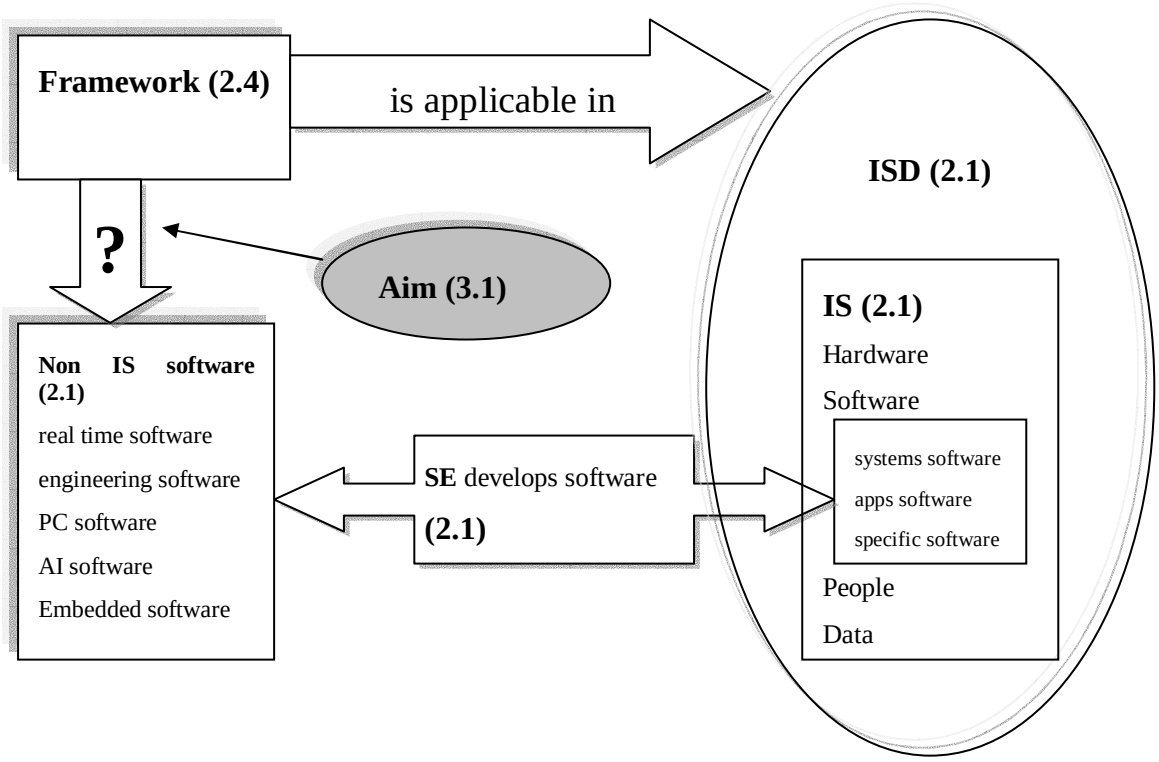

Figure 7: graphical summary of the problem area.

3.1 Formulation of the aim

Given the information presented in the previous section the following formulation for the aim of the report can be deduced:

Does the framework show potential for application in software projects where the purpose is not to build an organisational information system?

When considering software projects where the purpose is not to build an organisational IS this report refers to such software projects where the purpose is to build: real time software, scientific and engineering software, embedded software, PC software or AI software.

In order to fulfil the aim important considerations need to be made. The framework has been established through an extensive literature analysis within the areas of ISD and organisational development. This analysis has resulted in the identification of a number of success factors which have subsequently been grouped into the categories: boundary, objective and stakeholder. If the framework is to be considered for use in software projects where the purpose is not to build an organisational IS then similar

information must be gathered by the author of this report which concentrates specifically on non-IS software projects and the success factors related to them.

The investigative undertone of the formulation is important here as this inflection lends itself immediately towards the establishment of facts through thorough inquiry. In other words, in order to fully satisfy the aim all considerations must be made. It will therefore be necessary to identify companies/organisations that conduct software projects other than the building of ISs. Once suitable companies/organisations have been identified their collaboration can only be realised through some kind of field work. Once the field work has been carried out an evaluation process must be carried out to interpret the gathered materials validity with respect to the aim. The results of this effort must then be presented in a written report.

From the preceding discussion 5 major objectives can be deduced. These objectives are deemed essential for the aim to be satisfied:

- *Which success factors characterise a software project where the purpose is not to build an organisational IS?*
- *Which companies conduct non IS software projects*
- *Carry out field work to gather information*
- *Evaluate the material gathered in the field work*
- *Complete the report*

The aim of the report has now firmly been established and five objectives have been identified. These objectives will be readdressed in chapter 4 where they will be discussed in connection with plausible methods.

3.2 Delimitations

In order to obtain the required information it is necessary to limit the study to software companies that manufacture the types of software identified in 2.1. Once suitable companies have been identified it is essential to identify and involve people who possess relevant knowledge about the concepts in question. This is motivated by the fact that the framework outlined in section 2.4 is founded primarily on success factors that cover a variety of topics and themes. Using this as a base it can be anticipated that any material gathered throughout the course of this report will also be of a varied character. In order to investigate the framework's use in software projects that are not ISD related it is essential therefore to identify and involve people that lead software projects and thus possess knowledge that is of a varied character. It is unlikely that technicians and programmers will possess this varied knowledge and the study will therefore focus on involving project managers.

3.3 Expected result

The investigation expects to reveal specific success factors regarding the development of software where the purpose is not to build an organisational IS. It is anticipated that these success factors may, in some cases, be identical to those already identified in the framework. It is further anticipated that differences will be apparent. This is expected as the domains of ISs compared to other software systems are different. The investigation further expects to be able to contribute with useful information which may be used when considering the applicability of the framework in software projects where the purpose is not to build an organisational IS.

4 Method

In this section a thorough presentation of the problem solving process will be presented. The aim of the report has been outlined in section 3.1. Figure 8 graphically portrays the current status of the report, a problem has been formulated and a scientific method, or several methods, will be applied in order to achieve a result.

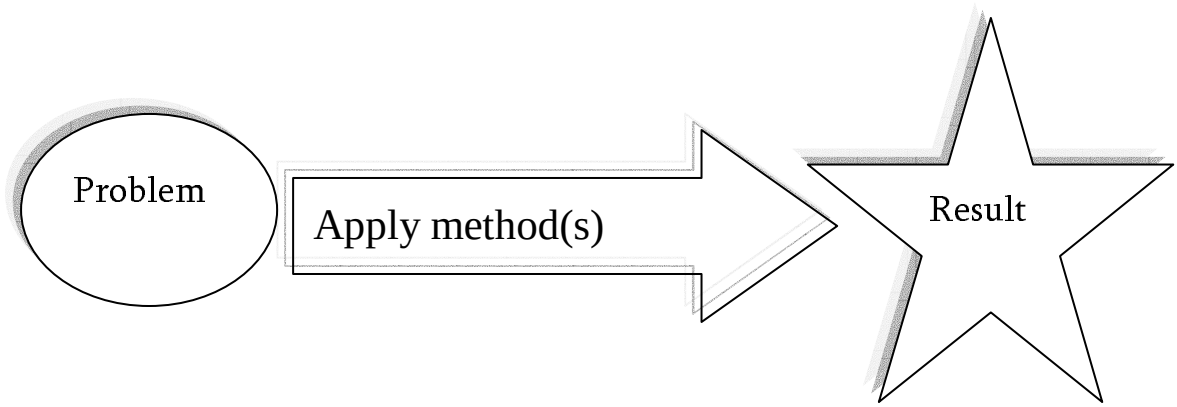

Figure 8: from problem to result

In section 4.1 a diagram of the anticipated workflow will be presented and discussed. By graphically presenting the problem solving process it is anticipated that a clear correlation between the underlying aim and its objectives can be clarified. In section 4.2 a diagram associating the underlying objectives with suitable methods by which to approach the formulated problem will be presented and discussed. A more in depth presentation of the identified methods will then follow. Section 4.3 concludes this section and will provide an account for the chosen method(s) and an explanation as to how these methods will be implemented in this report.

4.1 Identification of the anticipated activities

The purpose of this section is to account for the problem solving process by describing the steps which need to be taken in order for the aim to progress from a mere formulation through to a solution. Patel and Davidson (1994) advise that the formulated problem constitutes a base from which the forthcoming activities can be planned. Berndtsson, Hansson, Olsson and Lundell (2002) explain that in order to reach the overall aim of the report a number of objectives should be formulated. An objective is defined by Berndtsson et al (2002) as:

> *"a small achievable and assessable unit, i.e. a sub-goal of the project."* (Berndtsson et al, 2002, p. 55)

Once the objectives have been identified suitable methods can be then be associated with these objectives. Berndtsson et al (2002) refer to methods in this context as:

> *"a systematic endeavour to address a problem."* (Berndtsson et al, 2002, p.55)

Different methods may be associated with the different objectives (Berndtsson et al, 2002). This would indicate that a selection of methods may be a valid choice in order to achieve the overall aim of the report.

Dawson (2000) offers a structured approach for breaking down the problem solving process into different levels of detail thereby capturing the activities that need to be carried out in order to achieve the aim. Work breakdown structures (WBSs) according to Dawson (2000) enable a project to broken down into main objectives that have their origin in the formulated problem.

Figure 9 shows a WBS of the report's objectives based on the ultimate aim of the report, which is located at the root of the structure. In accordance with Dawson (2000) the first level after the root represents the five objectives of the report which can be compared to the objectives identified earlier in section 3.1.

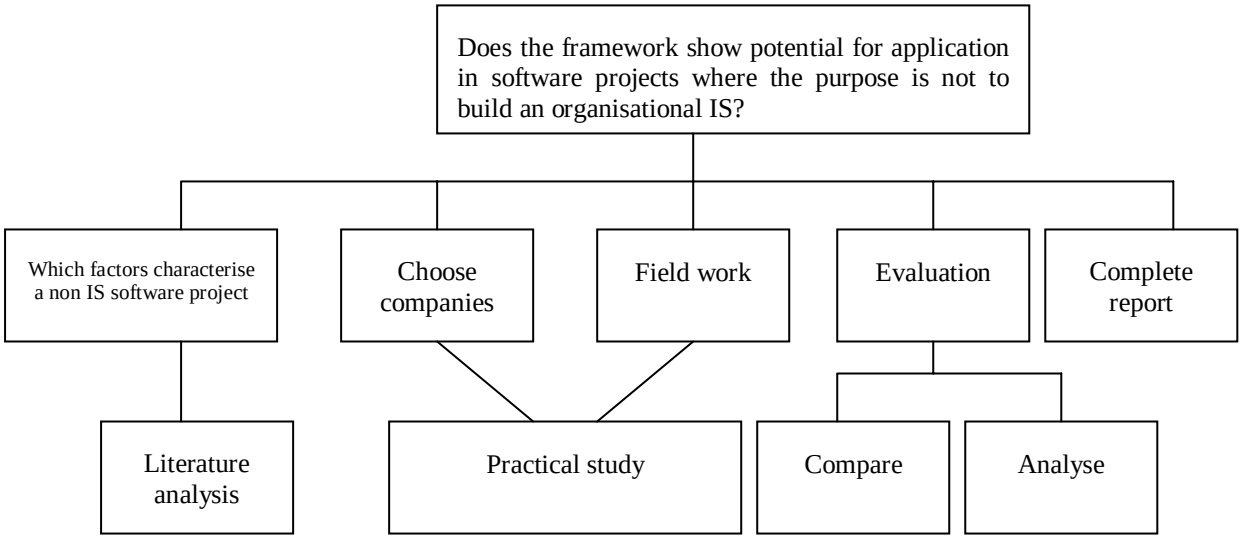

Figure 9: WBS of report objectives based on the aim

The five objectives represent the work that needs to be carried out in order to achieve the aim. It can be commented that the report is divided into two main areas of work: a *literature analysis* and a *practical study*. The practical study will encompass those activities that are concerned with the collecting of field data. When the gathering of field data has been completed an evaluation process will take place. The completion of the written report constitutes the final and indeed ultimate objective of this report.

4.2 Aligning methods to the practical study

In section 4.1 the five objectives of the report were proposed and two main area of work were identified. In figure 10 the practical study has been aligned to plausible methods. The coloured squares represent additions to the previous diagram (figure 9).

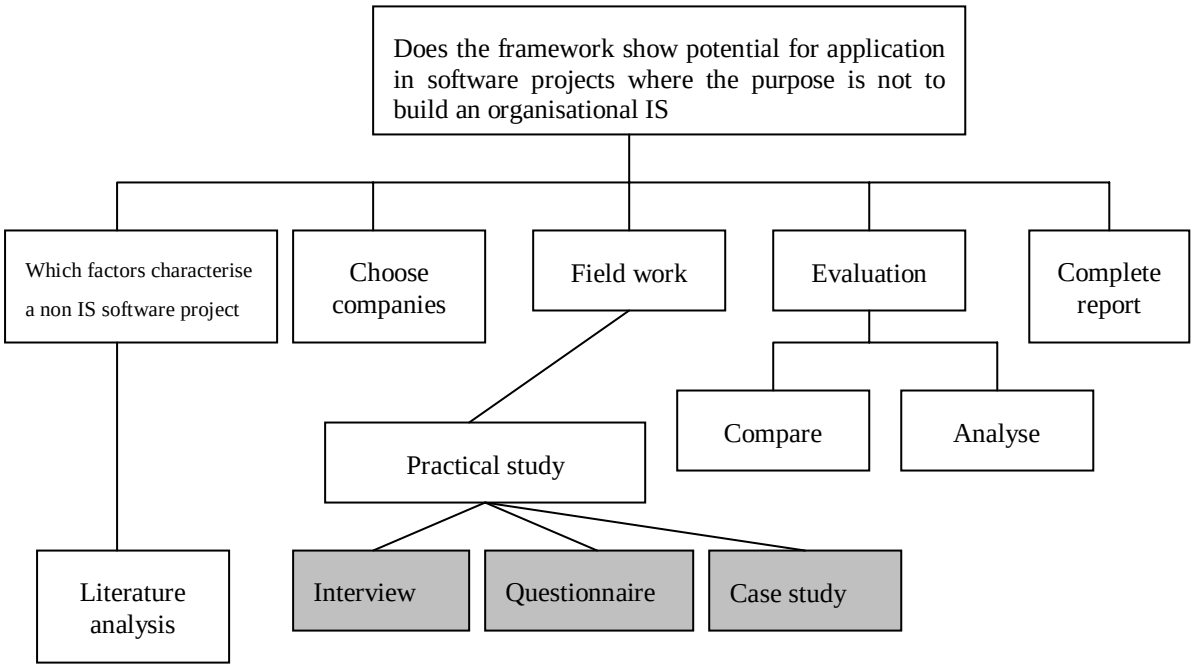

Figure 10: WBS aligning the practical study to suitable methods

In order to obtain a comprehensive, stable base from which a practical study can be conducted the author if this report feels that an examination of documented successful software projects (case studies) is a requirement. By performing such a study a better understanding of the dominating themes and factors highlighted by these case studies can be acquired. This material can then be incorporated into, or simply considered when planning the following stages of the practical study.

The aim of the report has been outlined in section 3.1. By studying the formulation of the aim it becomes apparent that some form of fieldwork is required to gather the required data. With this in mind it can be commented that a qualitative study may offer viable strategies by which the rest of the practical study can be realised. Repstad (1999) informs that qualitative studies are applicable in those situations where the focus is to obtain understanding for distinctive characteristics in a specific environment and how these characteristics have developed over time. This is in contrast to quantitative methods where according to Repstad (1999) the purpose is to ascertain how specific qualities are distributed amongst a number of people.

Berndtssson et al (2002) ascertain that qualitative methods have their foundation in social sciences where the primary concern is to obtain increased understanding of phenomena rather than providing an explanation for them. Quantitative methods according to Berndtsson et al (2002) originate from natural sciences where the objective is to obtain understanding for how something works. Quantitative research is hypothesis based were the ultimate goal of the research is to prove the hypothesis wrong.

The choice of a qualitative approach in the context of this report can be motivated through the fact that the underlying objective of the report is to obtain opinions, interpretations and additional information about the phenomena in focus. In simpler terms, the aim of this report is to obtain a deeper understanding of the phenomena in focus without necessarily providing an explanation for these phenomena. A qualitative study is therefore considered the only viable alternative by which such information can be obtained.

In a similar way the rejection of a quantitative approach in the context of this report can be motivated through the simple fact that the aim of the report is *not* hypothesis driven. It is not the intention to establish how something works and there are no predetermined variables.

4.2.1 Presentation of suitable methods

In figure 10 suitable methods were aligned to the practical study. A short discussion considering two plausible approaches concluded in the fact that a qualitative approach best suited the characteristics of this report. The methods that have been aligned to the objectives are of a qualitative nature and a presentation of these methods will now follow.

It can be commented that throughout the course of this section the terms method and technique will be used interchangeably. Berndtsson et al (2002) inform that there is little consensus about what counts as a method and what is understood to be a technique for data collecting.

It has been suggested that an examination of documented case studies could initiate the practical study. It has further been suggested that these case studies could provide valuable information when planning the rest of the practical study. The examination of documented material leans towards some kind of literature survey or literature analysis. In figure 10 this plausible method has been linked to the objective: *which factors characterise a non IS software project* and thus provides an excellent starting point for this section.

Literature analysis and literature survey

According to Dawson (2000) a literature survey comprises two main components: a *literature search* and a *literature review*. Dawson (2000) informs that a literature search involves the searching of and managing of research material whereas a literature review involves the understanding, critical evaluation and presentation of the obtained material.

Berndtsson et al (2002) refer to a *literature analysis* as the systematic analysis of a problem by analysing published sources. A literature analysis according to Berndtsson (personal contact, 10 mars, 2003) is more concerned with the analysis of literature while a literature survey is more concerned with compiling an overview of the literature. Berndtsson (personal contact, 10 mars, 2003) advises that the terms literature survey and literature analysis, depending upon the area of research and its praxis, can be used interchangeably.

By studying documented case studies of successful and unsuccessful software projects a deeper understanding of the factors concerning software project success can be realised. This knowledge would in turn provide a stable base on which a deeper study could be built.

Advantages and disadvantages

Dawson (2000) observes that journal papers and conference papers provide the latest findings in a specific field

Berndtsson et al (2002) outline the difficulties one can anticipate when undertaking a literature study: *completeness* and *identification of relevant sources.* Completeness according to Berndtsson et al (2002) is knowing when enough material has been gathered. Identification of relevant sources is self explanatory.

Interviews

Patel and Davidson (1994) observe that interviewing is an information gathering technique built on the formulation of questions. Frey and Oishi (1995) inform that an interview is a purposeful conversation. In simple terms an interview according to Frey and Oishi (1995) involves one person (the interviewer) asking prepared questions to another person (the respondent). Patel and Davidson (1994) refer to the *degree of standardisation* and the *degree of structure* when considering interviewing as a technique for the gathering of information. The degree of standardisation according to Patel and Davidson (1994) refers to the thoroughness of the formulated questions and the order in which they will be presented. The degree of structure is referred to by Patel and Davidson (1994) as the extent by which the interview questions can be interpreted by the respondent.

Berndtsson et al (2002) observe that interviews can be undertaken in a variety of ways and that there are different types of interview: *open interviews and closed interviews.*

Berndtsson et al (2002) explain that open interviews are commonly used in qualitative research and are characterised by the fact that the researcher has limited control of the issues raised during the interview session. According to Patel and Davidson (1994) open interviews are characterised by their low degree of standardisation and structure which in a real-world context would imply that interview questions are formulated on the spot and in such a way as to provide the broadest possible scope for interpretation.

A closed interview according to Berndtsson et al (2002) consists of a set of fixed questions that the interviewer asks during the session. In its purest form this type of interview does not allow additional information to be added during the course of the session. Patel and Davidson (1994) observe that this type of interview is completely structured and standardised. The interview questions are formulated in advance and do not offer a broad scope for interpretation. The questions are asked in exactly the same order to each respondent.

As is highlighted by Frey and Oishi (1995) an interview is a purposeful conversation. An interview in the context of this report is considered to be a valid method by which people involved in software projects can be contacted and given the possibility to share their experiences and opinions regarding such projects.

Advantages and disadvantages

High quality data: Judd, Smith & Kidder (1991) maintain that the response rate for interviews is high, up to eighty percent. Judd et al (1991) explain that in a face-to-face situation the interviewer can establish a rapport with the respondent thus motivating this person to give complete and detailed answers.

The presence of recording equipment can be a distraction: Repstad (1999) observes that the use of recording apparatus during an interview can heavily impede the result

due to the fact that some respondents display a somewhat allergic reaction to the thought of being recorded.

Questionnaire

Andersen and Schwencke (1998) explain that a questionnaire is a data gathering technique that involves the sending of a form consisting of predetermined questions to preselected people. Patel and Davidson (1994) add that although questionnaires are usually acknowledged as being sent by post or electronically it is possible to undertake a questionnaire based study under "supervision" where the interviewer meets the respondent face to face. Andersen and Schwencke (1998) inform that a questionnaire can be structured in different ways: *fixed questions* or *open questions.*

Fixed questions are often formulated with the intention of being easy to answer. It is not uncommon for fixed questions to be formulated in accordance with specific alternatives, albeit a scale or a simple yes or no (interpreted from Andersen and Schwencke, 1998). Open questions are, on the other hand, formulated with the intention of gathering a deeper, fuller answer from the respondent and therefore often require a written response (interpreted from Andersen and Schwencke, 1998).

Patel and Davidson (1994) inform that questionnaires formulated using fixed questions are completely structured whereas, depending on the formulation, open questions imply a lesser degree of structuring.

A questionnaire based study does not suffer geographical constraints and therefore provides greater access to individuals than what is possible through interviews. As is the case with interviews, a questionnaire is a question-based information gathering technique (see e.g. Patel and Davidson, 1994). By sending a questionnaire electronically, a large number of individuals involved in software projects can be reached which, as is the case with interviews, has potential to generate valuable information about the phenomena in focus.

Advantages and disadvantages

Large geographical spread: Andersen and Schwencke (1998) observe that a questionnaire has a large geographical scope and can be used to reach people that otherwise may be difficult to contact

Low response frequency: Berndtsson et al (2002) advise that a low response rate is a inherent difficulty with questionnaire based studies. Judd et al (1991) explain that if only twenty percent of the sample population respond to the questionnaire then there is no way of knowing if the characteristics of this sample can be generalised to describe the whole sample population.

Case study

A case study according to Berndtsson et al (2002) is an in depth exploration of a phenomenon in its natural setting. Berndtsson et al (2002) explain that a case study may, for example, be carried out in an organisation where the purpose is to explain and understand something within the organisation. Individuals or groups of individuals according to Berndtsson et al (2002) may also constitute suitable objects on which a case study may be carried out.

Patel and Davidson (1994) inform that a case study implies a study constrained by the fact that it is carried out on a limited group. A case study according to Patel and Davidson (1994) is characterised by the holistic perspective which is adopted with the purpose of acquiring as much coverage as possible. In accordance with Berndtsson et

al (2002), Patel and Davidson (1994) agree that case studies can be carried out in an organisation, on specific individuals or groups of individuals.

A case study offers the possibility to study phenomena in its natural environment. In the context of this report this would imply that the stages in a software project could be monitored and thereby access to valuable information regarding the stages in the development process and the decisions which are made could be obtained.

Advantages and disadvantages

*Broad coverage is obtained***:** as is highlighted by Patel and Davidson (1994) a case study adopts a holistic perspective in order to obtain as much coverage of the studied phenomenon as possible.

Generalisation can be difficult: Patel and Davidson (1994) explain that generalisation is the term used to describe how representative the obtained result is for those individuals who were not included in the study. According to Patel and Davidson (1994) depending on how the cases are chosen is dependent on the generalisation effect.

4.3 Choice of methods

In section 4.2 plausible methods were aligned to the practical study. A presentation of these methods was given along with their advantages respective disadvantages. In this section an explanation will be provided for the chosen methods. Emphasis will be placed on how these methods will be applied in order to obtain the required information.

Dawson (2002) identifies five elements that need to be controlled as the report progresses:

- Time
- Cost
- Quality
- Scope
- Resources

These factors according to Dawson (2000) are related to one another in the sense that it is important to weigh these factors against one another in relation to the problem in focus. It is the opinion of the author of this report that by disregarding these factors is to invite unnecessary complications and therefore these factors are considered entirely relevant when considering which method(s) to choose.

4.3.1 Alignment of report objectives to methods

In figure 11 the chosen methods have been aligned to the report's objectives. All new additions have been coloured.

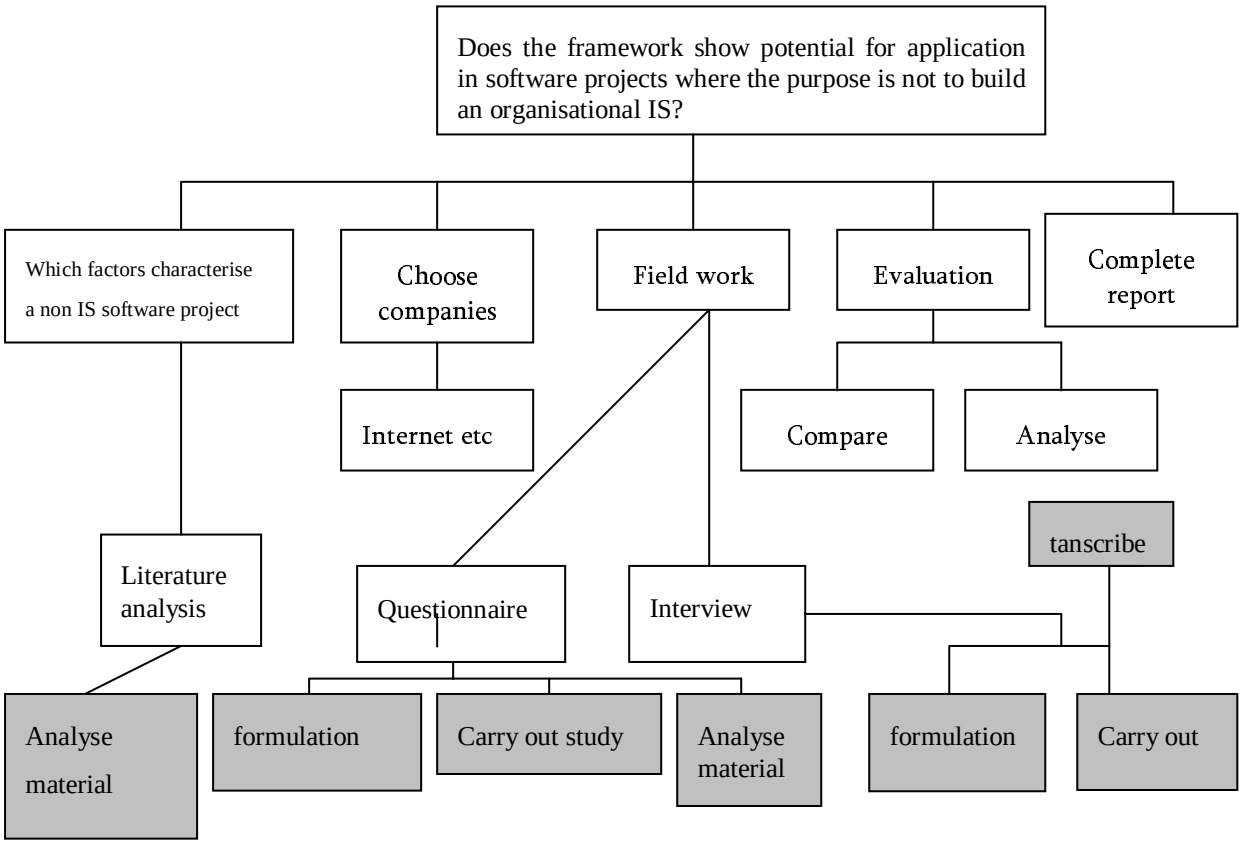

Figure 11: WBS of the data gathering process

Figure 11 graphically portrays the workflow of this particular report. Methods have been aligned to the objectives: *which factors characterise a non IS software project* and *field work* and a lower level of detail has been captured which serves to represent the stages associated with the identified methods. The breaking down of the WBS into lower levels of detail is in accordance with Dawson (2000) who advises that this activity is desirable but must also be regulated. A report according to Dawson (2000) should not be consistently broken down into tasks that take less than five percent of the allocated report time to complete. As a consequence of this figure 11 represents all the anticipated stages of this report and will not be broken down further.

Choice of method for examining documented case studies

An examination of documented literature requires some kind of a literature analysis. The author of this report feels that no further motivation with regard to this is required as a literature analysis of documented case studies is necessary in order to establish a stable base from which a deeper study can be founded.

Choice of method for choosing companies

The choosing of companies, in the context of this report, is something which can not be accomplished through an explicit method. The author of this report feels that the choosing of suitable companies with the purpose of partaking in a study is something that must be initiated by the author himself. The internet provides an excellent medium by which suitable companies can be identified. Additional sources are: personal contacts, recommendations from lecturers and the yellow pages.

Choice of method(s) for carrying out the field work

In figure 10 the objective: *field work* was aligned to a practical study. The practical study has in turn been associated with the plausible methods*: interview, questionnaire* and *case study*. It is anticipated that the field work will constitute the largest portion of the information gathering process and therefore the factors outlined by Dawson (2000) should be considered.

Questionnaire

A questionnaire is an information gathering technique based on questions (see e.g. Andersen and Schwencke, 1998; Patel and Davidson, 1994). A questionnaire is not constrained geographically and therefore presents a very attractive possibility for the gathering of information from a large population. Patel and Davidson (1994) inform that the term population refers to the delimited group a particular study is concerned with. For the purpose of this study the term population is adopted in this context and does not refer to, for example, the population of a country.

When considering the factors outlined by Dawson (2000) it can be ascertained that the factors time, quality and cost are paramount when considering the use of questionnaires. As is highlighted by Andersen and Schwencke (1998) questionnaires take a lot of time and work and therefore require careful thought with regard to how the gathered material will be processed. Depending on how the questionnaire is structured heavily influences the time it will take to process the information (Andersen and Schwencke, 1998). As time is a vital factor in this study it is intended that the literature analysis and structuring of the questionnaire will be executed sequentially. Dawson (2000) recommends the use of a simple graphical notation known as an *activity -on- the-node* diagram when planning the execution of tasks sequentially. Figure 12 portrays an activity-on-the-node diagram where the analysis of the information gathered during the literature analysis is conducted sequentially with the formulation of the questionnaire. The completion of these two tasks thus results in a complete questionnaire which can be sent.

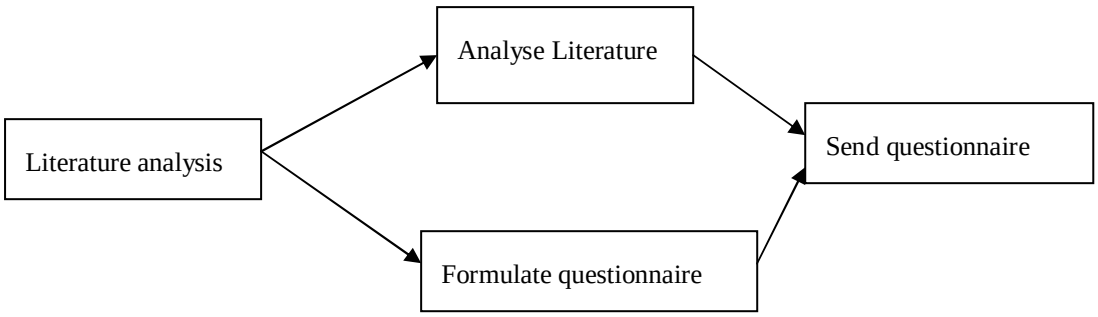

Figure 12: activity-on-the-node diagram of sequential tasks

Andersen and Schwencke (1998) observe that questionnaires formulated using fixed answers are easier to process than those questionnaires which are formulated using open questions. Patel and Davidson (1994) refer to the degrees of structure and standardisation when considering question-based methods. In order to effectively use

the allocated time for this study the questionnaire in this report will be constructed with the purpose of maintaining a high degree of standardisation. This implies that fixed questions will be formulated from the information obtained from the literature analysis. Fixed questions are often formulated with the intention of being easy to answer. It is not uncommon for fixed questions to be formulated in accordance with specific alternatives, albeit a scale or a simple yes or no (interpreted from Andersen and Schwencke, 1998). According to Patel and Davidson (1994) varying these alternatives is a requirement for motivating the respondent. It is anticipated that not all questions can be satisfactorily answered by specific response alternatives. With this in mind follow up questions may be included which will serve to invite the respondent to elaborate on his/her opinion regarding a particular topic.

It has been suggested that quality is a vital factor to be considered when using a questionnaire. Motivation of the respondent when using questionnaires is addressed by Patel and Davidson (1994) who advise that motivating the respondent is essentially reliant on the quality of the missive or covering letter which accompanies the questionnaire. According to Patel and Davidson (1994) a missive should include all relevant information and should be correctly formulated. The questionnaire in this study will be accompanied by a missive which will aim to be as stimulating as possible.

As is highlighted by Judd et al (1991) a major advantage with questionnaires is their low cost. In a report such as this where resources are limited such a factor plays a vital role when considering choice of method.

Interview

As is the case with a questionnaire, an interview is also an information gathering technique based on questions (Patel and Davidson, 1994). It has been established that the literature analysis and formulation of the questionnaire will be carried out sequentially (see figure 12). It is intended that the information gathered from the questionnaire will then provide a base from which interview questions can be formulated.

The purpose of the questionnaire is to reach as broad a scope of respondents as possible. This is not achievable with interviews (see e.g. Andersen and Schwencke, 1998) and instead the emphasis will be on obtaining time with a few key people in software organisations.

Berndtsson et al (2002) explain that in an open interview the researcher has little or no control over the issues that will be raised during the interview session. A closed interview according to Berndtsson et al (2002) is characterised by specifically formulated questions where the researcher has full control of the issues raised. The purpose of interviews in the context of this report is to motivate a deeper discussion about those issues raised by the material gathered from the questionnaire and therefore a relatively low degree of standardisation and structure will be sought after. In accordance with Patel and Davidson (1994) this would imply that questions may be formulated during the actual interview session (*degree of standardisation*) and that the questions are formulated in such a way as not to be answerable by a simple yes or no (*degree of structure*).

Andersen and Schwencke (1998) recommend the use of an interview guide when conducting an interview session. An interview guide according to Andersen and

Schwencke (1998) is a list of those questions or topics that will be raised during the session. This mechanism is considered by the author of this report to be an effective way of structuring the interview and will therefore be adopted.

According to Judd et al (1991) an advantage with interviews is the high quality data which can be obtained. When considering the factors outlined by Dawson (2000) it becomes apparent that quality, cost and time are of importance when considering the use of interviews. The processing of the information gathered during the interview sessions is a time consuming task (Repstad, 1999). The material captured on tape during the interview sessions must be transcribed. As a guideline Repstad (1999) comments that a one hour interview corresponds to approximately 20-25 pages of text. It is anticipated that the interviews undertaken in this report will not exceed 30 minutes and will be limited to four sessions. This corresponds to, using Repstad (1999) as a guideline, approximately 40 pages of text.

4.3.2 Summary

The purpose of this chapter has been to relate the aim of the report to suitable methods by which a result can be achieved. Figure 12 outlines which methods will be applied in order to achieve the result and thus satisfy the aim.

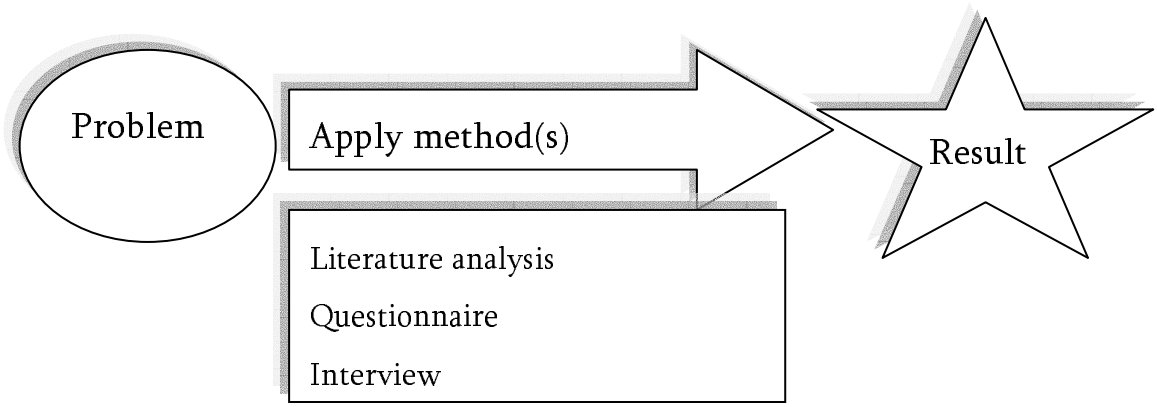

Figure 12: methods by which to achieve the result

Through the use of WBSs (see Dawson, 2000) the report has been broken down into objectives. From these objectives two principal areas of work have been identified: *literature analysis* and *practical study*. The practical study has subsequently been aligned to suitable methods. It is anticipated that the literature analysis will provide valuable information from which a questionnaire can be based upon. The information gathered from the questionnaire is in turn intended to be used as a foundation for conducting interviews which will aim to motivate a deeper discussion on those issues raised by the questionnaire.

4.4 Reflections over the chosen research methods

Berndtsson et al (2002) inform that *validity* and *reliability* are important aspects to consider when choosing which method(s) to apply to a problem. Berndtsson et al (2002) explain that validity concerns itself with the relationship between what you intend to measure and what you actually measure. Reliability concerns itself with the accuracy of the method. Furthermore, methods are only reliable and valid within a range of uses.

In order to fulfil the aim of this report three methods have been chosen: literature analysis, questionnaire and interviews. It is the opinion of the author of this report that these methods are valid and reliable within the range of the problem area; i.e. they can effectively be implemented with the purpose of investigating the applicability of the framework in software projects where the purpose is not to build an organisational IS.

Patel and Davidson (1994) explain that when interviews are employed as a method for gathering data the reliability of this method is heavily dependant upon the interviewer's ability. Patel and Davidson (1994) mean that the interviewer should use judgement when registering the responses and if the interviewer is not experienced in this then mistakes can occur. Patel and Davidson (1994) advise that relatively good reliability can be obtained if standardised interviews are employed. A further threat to reliability in the context of interviews is the *interview effect* which can occur. (Andersen and Schwencke, 1998; Patel and Davidson, 1994) explain that the interview effect is that effect which the interviewer has on the respondent.

As has been highlighted, the purpose of interviews in the context of this report is to motivate a deeper discussion about those issues raised by the material gathered from the questionnaire and therefore a relatively low degree of standardisation and structure will be sought after. This raises immediate issues concerning the reliability of the chosen approach and will be readdressed in chapter 8.

5 Carrying out the study

In this section an account for the data gathering process will be given. This is best accomplished by allocating a section to each of the chosen methods.

5.1 Literature analysis

The use of a literature analysis in the context of this report has been motivated by the fact that the information obtained from such a study would provide a stable base from which a further, deeper study could then be conducted. By analysing case studies of successful software projects it was anticipated that these documents had the potential to reveal specific characteristics which could then be taken into consideration when formulating a questionnaire.

The initial starting point for the literature analysis was the bibliographic databases which could be accessed through the university library's homepage. In accordance with Berndtsson et al (2002) a browsing strategy using keywords specific to the formulated problem was adopted. Examples of such keywords were: case study, software project, success factor(s), critical success factor(s), lessons learned etc. These keywords were obviously used in combination and yielded a satisfactory yet limited response. As a consequence of this the scope of the search was broadened to incorporate scientific journals, books and a variety of non-scientific publications.

The search for specific case studies yielded a disappointing result where insufficient material was retrieved. By increasing the level of abstraction and adjusting the search variables a wider range of material was retrieved. Berndtsson et al (2002) advise that completeness and identification of the right sources are inherent problems with literature analyses (see section 4.2.1). In an attempt to overcome this obstacle a high level of abstraction was sought after.

Procaccino et al (2002) identify seven chief categories which are considered influential to software project success: management, customers and users, requirements, estimation and scheduling, the project manager, the software development process, development personnel. This high level of abstraction provided an excellent mechanism by which to constrain the literature analysis and thereby increasing the probability of identifying relevant sources. The latter point would be in accordance with Berndtsson (2002) who advise that this can pose a potential problem when conducting a literature analysis.

By incorporating these seven categories into the browsing strategy and combining them with the previously identified keywords i.e.: management of software projects or users in software projects it was possible to obtain further information thus strengthening the identified material.

The information that was gathered by the literature study was critically reviewed, dissected and presented in section 6.1. The literature analysis identified a number of success factors relevant to software project success. For the purpose of clarity these success factors have been presented in table form (see table 2, sub-section 6.1.3)

5.2 Constructing the questionnaire

In section 4.3.1 it was established that the literature study and the formulation of the questionnaire would be carried out sequentially. The categories identified by Procaccino et al (2002) in the previous sub-section provided an excellent framework

by which to divide up and structure the questionnaire. Patel and Davidson (1994) inform that thorough preparation is the norm for both questionnaires and interviews. As was highlighted in sub-section 4.31:

> *"in order to effectively use the allocated time for this study the questionnaire in this project will be constructed with the purpose of maintaining a high degree of standardisation."* (p. 37/38)

An explanation for standardisation has already been given (see sub-section 4.2.1). In order to achieve this, fixed alternative questions were formulated based on the success factors obtained from the literature analysis. Patel and Davidson (1994) recommend that when formulating fixed alternative questions it is important to vary the response alternatives in order for the respondent to maintain his/her motivation. Patel and Davidson (1994) offer the following response alternatives:

- Response alternatives which give either-or, for example YES/NO
- Response alternatives which include frequencies, amounts etc
- Response alternatives in list form
- Response alternatives which should be ranked
- Response alternatives associated to a scale or a grade

These alternatives were considered and in some cases incorporated into the questionnaire (see supplement 2). Where the response alternatives were given as YES or NO the respondent, when answering YES, was encouraged to answer a follow up question inviting the respondent to explain *why* this was the case. According to Patel and Davidson (1994) why questions are suitable as follow up questions.

When formulating the questions to be included in the questionnaire it is important to acknowledge certain aspects. Andersen and Schwencke (1998) offer the following guidelines when polishing up the questions to be included in the questionnaire:

- The questions should be formulated clearly and concisely
- The response alternatives should cover all available alternatives
- It is easier to answer questions that are mutually exclusive, only one alternative should be marked
- Avoid formulating the questions in such a way that *leads* the respondent towards a certain answer

Questions aimed at gathering opinions related to a particular statement were also included into the questionnaire. Patel and Davidson (1994) refer to such questions as attitude based where the purpose with such questions is to establish what a person thinks about a certain phenomena. In order to formulate these questions a mechanism known as the *Likert –scale* was used. This mechanism is covered in Patel and Davidson (1994, p. 71) and was adapted for the purpose of this report. As the attitude based questions served purely to establish an opinion concerning a specific statement only two response alternatives were used.

Once the questions were formulated (see supplement 2) the questionnaire was sent to randomly chosen software companies. All questions in the questionnaire were formulated in Swedish. This is explained by the fact that the questionnaire survey was carried out in Sweden and thereby a decision was made to confine the geographical

scope of the survey to Sweden. In the interest of generating a positive response it seemed natural to adopt the main language of potential respondents.

Once the questionnaire was formulated a missive (covering letter) was written (see supplement 1). The missive was also formulated in Swedish and was sent as an e-mail, with the questionnaire as an attachment, to all the companies chosen to take part in the survey. All questionnaires were sent electronically. Potential respondents were advised that they could respond electronically or alternatively via normal post. Confidentiality of all material was ensured.

The companies chosen to take part in the questionnaire survey were chosen randomly from the internet. Using the criteria outlined in chapter 3 a decision was made to confine the survey to smaller software companies who manufactured any of the following types of software: real time software, PC software, embedded software, AI software or engineering software. It was anticipated that the response would be greater from these smaller companies. This is motivated by the fact that in smaller companies the chance of the questionnaire reaching the right person was considered to be greater than what it would be if the questionnaire were sent to a large software house or software development organisation.

In total 30 questionnaires were sent out. On the first day four responses had been received. Two companies advised that they did not have time to partake in the survey. One company was not involved in software development and one company answered and returned the questionnaire electronically. On the last day of the survey the response rate remained at one returned questionnaire. A reminder was sent via e-mail to the 26 remaining companies and the response time was extended by two days. Ten answers came in the same day advising that due to work volume no-one was available to answer the questionnaire. The material generated by the questionnaire corresponded to a response rate of 3% which is deemed inadequate for any kind of analytical purpose.

One can speculate back and forth as to why the questionnaire based study failed so miserably. It is however important to bear in mind that undertaking a questionnaire based study does assume understanding and acceptance of the risks that accompany this particular technique. The probability of a low response rate is a well documented risk factor.

In the context of this report a plausible theory as to why the response rate was so low may be found in the choice of companies. As has been explained, a decision was made to confine the survey to smaller companies under the assumption that the questionnaire would have a greater chance of arriving at the right person's desk. Although this may have been the case these people subsequently did not have time to fill in the questionnaire due to the sheer volume of work their respective companies were experiencing. If the questionnaire had been sent to larger companies who inevitably employ more people then, on reflection, the chances of someone having time to fill in the questionnaire are considerably greater.

An additional failure factor could be the length of the questionnaire. Although the questions have been formulated using a variation of fixed alternatives, which usually require less time to answer than open questions, the respondent may have been scared off by the length of the questionnaire and not reflected over how much time would actually be required to fill it in.

5.3 Preparing for the interviews

This section will account for the work carried out with regard to the interview sessions that have been conducted in this report. As was established in section 4.3.1:

> *"The purpose of interviews in the context of this report is to motivate a deeper discussion about those issues raised by the material gathered from the questionnaire"* (p. 38)

The initial intention was that the questionnaire would provide a foundation from which a further interview based study could be built. It has been highlighted in section 5.2 that the low response rate generated insufficient material and as a consequence the questionnaire has been omitted from the investigation. On the basis of this the interviews would now provide the only means by which opinions on the topics raised in the literature study could be discussed with software professionals.

In early march, before the questionnaire was sent out, an e-mail was sent to contact people at two local software companies, company X and company Y, where a request to interview two project managers from each company was expressed. The purpose of the interviews was explained along with the expected duration of the interviews. A simplified account for the topics that were likely to be addressed was also given. Confidentiality of all material was also ensured. It was further outlined that the potential respondents should have experience in managing software projects, the software in question being compatible to those types outlined in chapter 3: real time software, PC software, AI software, embedded software, engineering software. Additionally, it was highlighted that the intention was to record the interviews with help of a tape recorder.

On the basis of this e-mail four respondents were identified and interview sessions were booked accordingly. These respondents are represented in table 2. Three of these sessions were pencilled in for week 16 and the fourth for week 17. Due to other engagements the fourth interview was then moved to week 18. The search for potential respondents was confined to the local area as it was intended that the respondents would be met in person.

R1	Male project manager: company X
R2	Male project manager: company Y
R3	Male project manager: company X
R4	Female project manager: company X
R5	Male project manager: company Y

Table 2: presentation of the respondents

During the course of week 15 when it became apparent that the questionnaire was heading for failure a further interview was booked at company X. An attempt to identify further respondents at company Y proved fruitless. The person from the software company who answered the questionnaire was also contacted with the intention of conducting a telephone interview but again this proved fruitless. Two additional software organisations in the Gothenburg region were contacted but were unable to assist.

When forming the actual structure of the interviews a decision was made to adopt an open question approach. Berndtsson et al (2002) explain that in an open interview the researcher has little or no control over the issues that will be raised during the interview session. This approach is motivated through the fact that open questions offer the opportunity for the respondent to freely interpret the scope of the question thus enabling a richer answer. Patel and Davidson (1994) refer to this aspect as the degree of structure, something which has been addressed in sub-section 4.2.1.

Patel and Davidson (1994) highlight four aspects which should be avoided when formulating interview questions:

- Long questions
- Leading questions
- Negations
- Double questions, for example: do you usually stay at home on your holiday or do you travel abroad.
- Anticipatory questions, for example: have you stopped drinking
- Why questions: these questions can lead to categorisation problems and thereby loss of information.

These aspects were considered by the author of this report to be particularly useful as they provide a simple set of easy to remember guidelines which can easily be implemented. Andersen and Schwencke (1998) recommend the use of an interview guide when conducting an interview session. An interview guide according to Andersen and Schwencke (1998) is a list of those questions or topics that will be raised during the session. An interview guide was constructed and used during the interviews carried out in this report (see supplement 3). The interview guide used in this report merely outlines the topics to be addressed. All questions related to these topics were formulated on the spot. The interview guide has also been structured in accordance with the seven categories highlighted earlier. Structuring the interview guide in accordance with these categories offers an efficient mechanism by which to both carry out the interview and compile the data gathered from the interview.

At the end of week 15 an additional E-mail was sent out to the four respondents who were to be interviewed during week 16. This E-mail better described the topics that would be taken up thus giving the respondents time to prepare if they felt this were necessary. At the end of week 17 a similar E-mail was sent to the respondent to be interviewed in week 18.

All respondents were met at their place of work. Before the interview session commenced the purpose of the interview was once again explained. Permission to record the information was obtained from all respondents. Confidentiality of all material was assured as well as the respondent's anonymity. The duration of the sessions varied between 30-40 minutes. Each respondent was thanked for his/her collaboration after each session

Repstad (1999) highlights that the processing of the information gathered during the interview sessions is a time consuming task. Each session was transcribed (transferred to paper) directly after its completion. According to Andersen and Schwencke (1998) this approach is tiring but gives many advantages, for example: ones memory of the actual session is better directly after its completion. The material obtained from each interview varied between 5-7 pages per interview. Once all material had been

transcribed the respondents were contacted, again via E-mail, and informed that a copy of the transcriptions could be sent to them if they so desired.

The interview material was critically reviewed and compiled. A presentation of this material can be found in section 6.2.

6 Material presentation

The purpose of this section is to account for the material that has been gathered by applying the methods outlined in chapter 4. This material will provide the foundation for chapters 7 where an analysis of the material and final result are presented.

In can be commented that the material generated by the questionnaire was limited to such an extent as to be deemed inadequate for any kind of analytical purpose in the context of this report. The material generated by the questionnaire will not be presented in this chapter and consequently will not be referred to in chapter 7.

In section 6.1 the information gathered through the literature analysis will be presented and in section 6.2 the interview material will be presented.

6.1 Presentation of material gathered during the literature analysis

This section will account for the material gathered during the literature analysis. As was established in chapter 5, a high level of abstraction has been acquired where seven major categories have been identified. These categories are deemed relevant when considering software project success. A discussion concerning these categories is presented in the next sub-section.

6.1.1 A presentation of seven categories for software project success

Procaccino et al (2002) provide a high level of abstraction when considering software project success. Seven categories are identified by Procaccino et al (2002 as being influential for a software projects success: management, customers and users, requirements, estimation and scheduling, the project manager, the software development process, development personnel. Three of these categories are acknowledged by The Standish group who in Procaccino et al (2002) highlight the following as being of crucial importance to software project success:

- *Managemen*t: inadequate management practices have far reaching implications for software development. A project needs a committed sponsor or champion throughout.
- *Customers and users:* research suggests that one of the most important contributors to successful project development is end-user involvement. This involvement should occur in all phases of the development process.
- *Requirements:* a clear understanding of the problem that assists in the production of well defined requirements is essential to the development of a successful system.

The three major categories outlined above would appear to be endorsed by Duvall (1995) who identifies problematic situations in software projects as being: requirements, developer, enabling technologies, change, customer, constraint and performance. Although Duvall (1995) identifies six categories it is the opinion of the author of this report that the groupings: *change, constraint* and *performance* can be classified under the broader heading management. This is motivated through the fact that Duvall (1995) defines change as something which implies movement in a project. Such movements according to Duvall (1995) consequently need to be managed. Constraints are accounted for by Duvall (1995) as being restrictions or limitations that subsequently affect the management of a project, whereas performance is defined by Duvall (1995) as the fulfilment of managerial and technical requirements of the

software under development. Duvall (1995) advises that aspects such as function, cost and schedule can be classified under performance. As can be seen, all three of these definitions imply a relationship or affiliation to management.

It is apparent that the three major categories: management, customer and user and requirements are of specific importance when considering software project success. However, it is the opinion of the author of this report that these three categories can be built upon thereby increasing the scope of the analysis. This will serve to strengthen the overall concept of software project success. The author of this report feels that Procaccino et al (2002) effectively encompass the scope of a software project with the seven identified factors: management, customers and users, requirements, estimation and scheduling, the project manager, the software development process, development personnel. As can be seen, the three categories identified by the Standish Group have successfully been incorporated into Procaccino's (2002) categorisation and therefore the author of this report feels that this categorisation suggested by Procaccino et al (2002) offers a stable foundation from which to work from.

6.1.2 Presentation of success factors identified in the literature analysis

This sub-section will provide an account for the information that has been gathered with respect to the seven categories identified in the previous sub-section.

Management

Capers Jones (1995) inform that the use of poor management practices contribute considerably to software project failure. Johnson (2001) on a more positive note highlights that software project success rates are improving. According to Johnson (2001) one reason for this improved success can be explained by better skilled project managers with better management processes.

Purba, Sawh & Shah (1995) inform that the project objective involves a clear definition of the project and the rationale for undertaking the project.

Purba et al (1995) identify the projects scope as those areas which are impacted by the project. Purba et al (1995) advise that the project scope can be as limited as a department in an organisation or as broad as an international concern. Johnson (2001) considers time to be the enemy of all projects. According to Johnson (2001) by minimising the scope of the project time is reduced thereby increasing the chances of success. (Field, 1997, in Reel, 1999) attributes an ill-defined project scope to software project failure

(The Standish Group, in Procaccino et al, 2002) maintain that a project needs a committed sponsor or champion throughout. This sponsor should participate in the decision making process and encourage the same commitment from other stakeholders. (The Standish Group, in Procaccino et al, 2002) strongly advise that if no sponsor is available at the start of the project it is important that one is obtained as soon as possible. (Field, 1997, in Reel, 1999) considers lost sponsorship to be characteristic of software project failure. Johnson (2001) observes that projects that lack a staunch project champion can easily drift into a technological or political abyss.

Reel (1999) informs that few companies adopt a process for learning from past mistakes. Reel (1999) observes that if time is not taken to learn from past mistakes then the same mistakes will inevitably be repeated. Reel (1999) adopts the term post-

mortem analysis for that phase or process which serves to capture the mistakes and successes of the completed project.

Matheson and Tarjan (1998) refer to a particular case where a software project failed despite displaying all the attributes typically attributed to success: committed team, good funding, enormous market demand etc. According to Matheson and Tarjan (1998) one reason for this project's failure was the lack of information flows among parties critical to the venture's success. This observation would be corroborated by Purba et al (1995) who advise that because so many people are involved on a project the manager must ensure that relevant groups communicate appropriate information to each other. Purba et al (1995) inform that this involves establishing guidelines on information communication and documentation development and distribution.

Paulish and Carleton (1994) inform that software measurement is becoming integral to improving software development. Paulish and Carleton (1994) explain that the Software Engineering Institute (SEI) recommend the following core measures for software measurement. It should be commented that the actual measure is followed by the unit of measurement: defects (number of defects found per phase), product size (lines of code-LOC) effort (staff hours), actual schedule duration time (months per phase), estimated schedule duration time (months per phase).

From a technical point of view, Capers Jones (1995) advise that inadequate routines for quality control seriously impede software projects. As is highlighted by Capers Jones (1995) the finding and fixing of bugs is a time consuming task. Capers Jones (1995) maintain that these issues seriously affect the management of the software project as problems first become visible to clients and corporate executives far to late for recovery. Reel (1999) enforces the relevance of quality control in software development. Reel (1999) observes that the establishment of procedures for achieving high quality is necessary before other development begins. Developers proven in their capability to produce high quality code should be hired and invited to partake in regular code reviews. Capers and Jones (1995) would substantiate this claiming that failure to use design reviews and failure to use code inspections are factors characteristic of failed projects.

A summary of important factors that affect management of the software development process are given as:

- *Well defined project objective* (Purba et al, 1995)
- *Well defined project scope*(Purba et al, 1995; Field, 1997, in Reel, 1999)
- *Minimise the scope of the project* (Johnson
- *Find a project sponsor /champion* (Johnson, 2001; The Standish Group, in Procaccino et al, 2002; Field, 1997, in Reel, 1999)
- *Implement a post-mortem analysis* (Reel, 1999)
- *Ensure effective communication between all parties* (Matheson and Tarjan ,1998; Purba et al, 1995)
- *Incorporate effective software measurement into the process* (Paulish and Carleton, 1994; Capers Jones, 1995)
- *Routines for effective quality control* –design reviews, code inspections- (Capers Jones, 1995)

The software development process

Duvall (1995) explains that a software development process is a series of actions or procedures defined and/or followed by the managers and developers to produce a software product

The use of a methodology in the software development process is considered by Johnson (2001) who poses the question:

> "*Does having a formal project methodology increase success rates?"*(Johnson, 2001, p. 134).

According to Johnson (2001) using a formal methodology provides a realistic picture of the project and the resources committed to it. Furthermore, certain steps and procedures are reusable. As is highlighted by Johnson (2001) the tendency to reinvent the wheel is therefore minimised. Johnson (2001) refers to research carried out by The Standish Group who have shown that 46% of successful projects used a formal methodology compared to 30% of challenged or failed projects. It is the assumption of Johnson (2001) that this factor should increase chances of project success by approximately 16%.

Rakos (1990) advises that software tools such as fourth generation languages (4GLs) and database management systems (DBMSs) greatly reduce development time. Capers Jones (1995) suggest that the continuous of planning tools in the development process is conducive to project success.

A summary of important factors that address issues concerning the software development process are given as:

- *The use of a formal methodology* (Johnson, 2001
- *Use software tools* (Rakos, 1990; Capers Jones, 1995)

Estimation and scheduling

Glass (1998b) recognises the importance and centrality of the schedule in software projects. The schedule (interpreted from Glass, 1998b) ultimately governs the quality and thoroughness of the development process as it is the mechanism by which time is allocated to the specific phases of the development process. Carelessness in scheduling can ultimately choke the process rendering it impossible to realise. Unrealistic deadlines are considered by (Field, 1997, in Reel, 1999) as being a major pitfall in software development

The relevance of scheduling is further addressed by Giglio (1999) who advises that scheduling enables the organisation of tasks into a logical sequence. Giglio (1999) observes that schedules are strengthened with the addition of milestones which give developers targets to aim for and act as early warning devices for detecting problems. Reel (1999) would agree acknowledging that a large problem in software development is knowing where you are in your schedule. Reel (1999) continues, adding that if you don't know where you are in your schedule then adjustments cannot be made to bring the project back on line. Capers Jones (1995) maintain that failure to monitor progress or milestones and failure to use estimating tools are factors directly attributable to software project failure.

According to Rakos (1990) accurate estimation is a problem in software development. Rakos (1990) informs that one reason for this is due to the fact that estimations are

given when the extent of the problem is not fully known. Johnson (2001) sums this up concisely in a simple sentence:

> *"Estimating is just plain hard."* (Johnson, 2001, p.135)

According to Johnson (2001) the key to estimation is being realistic. IT managers according to Johnson (2001) often need to muster all their collective knowledge and experience to come up with estimates that reflect the true effort required.

A summary of important factors that address issues concerning estimation and scheduling in the software development process are given as:

- *Include milestones* (Giglio, 1999; Capers Jones, 1995; Rakos, 1990)
- *Set up realistic deadlines* (Field, 1997, in Reel, 1999; Glass, 1998b)
- *monitor progress*(Capers Jones, 1995; Reel, 1999)
- *Use estimating tools* (Capers Jones, 1995)
- *Realistic estimations* (Johnson, 2001)

Development personnel

Reel (1999) advises that building the right development team is reliant upon identifying the right people. Reel (1999) emphasises the importance of creating the correct mixture of competence. It may not necessarily be fruitful to construct a team consisting of top designers. As Reel (1999) highlights, having too many stars creates ego issues and distractions. Not having enough expert knowledge consequently burdens the team leaving it struggling. This last comment would be endorsed by Giglio (1999) who advises against the over reliance on one programmer as the primary developer.

In addition to building the right team Reel (1999) warns for attrition (people leaving) in software projects. Reel (1999) advises that attrition is a constant problem in software projects as knowledge and competence leaves the team when a person walks out the door. Reel (1999) affirms that a replacement may well be found but this person must then be quickly bought up to speed on the workings of the project.

Rettig (1990) advises that team participation in decision making has many advantages. Amongst other things Rettig (1990) acknowledges consensus decision making as being directly attributable to "solution ownership". The concept of ownership is addressed by Giglio (1999) who advises that if everyone has a stake in the project the chances of success are increased. Glass (1998a) addresses the issue of perceptions in software development. Glass (1998a) highlights that what can appear to be a failure can, depending on who you ask, equally be a success. Glass (1998a) refers to a study carried out by K. R. Lindberg who, after confronting software practitioners for their perceptions of an allegedly failed project found that these practioners considered the project to be a success due to the fact that they were given, amongst other things, the freedom to develop a "good design". In other words these software developers were given ownership over their design.

Krishnam (1998) ascertains that team factors such as personal capability and experience, personal motivation and coordination and communication among team members are critical factors in software development. This issue is addressed by Duvall (1995) who in a study of software management practices received the

following answer from one respondent when asked what characterised a good software engineer:

> *"A lot of experience and knowledge about the computer and software development."* (Duvall, 1995, p. 113).

Field, 1997, in Reel, (1999) would endorse the above quotation. The lack of people with appropriate skills is identified by (Field, 1997, in Reel, 1999) as being a major pitfall in software development.

Rakos (1990) addresses the issue of roles in effectively summarises the necessity of roles or areas of responsibility in the following quotation:

> *"The responsibilities for project management must be clearly assigned to specific individuals, or everyone will think it is some other person's responsibility. Nothing will get done!"* (Rakos, 1990, p. 3).

Constantine, in Rettig, (1990) would agree suggesting that the project acquires structure from the recognition of certain roles that are affiliated to tasks deemed critical to the success of a software team. Such tasks are identified by Constantine, in Rettig, (1990) as being: decision making, coordination, information management, application domain knowledge, technical analytical functions and critical feedback. Roles that can be generated from these tasks are identified by Constantine, in Rettig, (1990) as: technical leader, process leader, information manager, technical and process critic and domain expert.

A summary of important factors that address issues concerning the development team in the software development process are given as:

- *Build the right team* (Reel, 1999)
- *Keep attrition low* (Reel, 1999).
- *Allocation of roles to team members* (Rakos, 1990; Constantine, in Rettig, 1990)
- *Skills of the development team* (Krishnam, 1998; Duvall, 1995; Field,1997, in Reel, 1999))
- *Give the development team ownership*(Giglio, 1999; Rettig, 1990)

The project manager

Rakos (1990) explains that the major role of the project manager is to interface the project team to the outside world. It is the project manager who is responsible for reporting to clients, upper management and anyone else concerned in the project. In addition, the PM acquires all the resources to get the job done.

Johnson (2001) advises that an experienced project manager is a requirement for a successful software project. Johnson (2001) ascertains that as many as 97% of all successful software projects are led by an experienced project leader. Purba et al (1995) would agree highlighting that one attribute of a successful software project is a project manager with the right leadership, management and technical skills

Reel (1999) informs that the ability to make smart decisions separates successful project managers from failures. This is further endorsed by Purba et al (1995) who ascertain that a project manager who can identify issues, communicate them effectively and resolve then in a timely manner is a necessity for a successful project.

Procaccino et al (2002) advise that an underlying assumption of software project success is that the project manager has been given full authority to manage the project.

A summary of important factors that address issues concerning the PM are given as:

- *The ability to make good decisions* (Reel, 1999; Purba et al, 1995)
- *Experienced project manager*(Johnson, 2001; Purba et al, 1995)
- *Project managers have full authority to manage their project* (Procaccino et al, 2002)

Customers and users

Rakos (1990) explains that the user has the responsibility to provide the systems analyst(s) with timely, reliable information. Rakos (1990) continues highlighting that it is a requirement that the user knows all about the existing system and what is required of the new one. In addition to these underlying qualities the user according to Rakos (1990) must be available and be authorised to make decisions about the system and how it will effect the organisation. Reel (1999) advises that whenever possible customers and users should be involved in the development process. According to Reel (1999) this will not only help to establish trust between the developers and users but will place domain experts within easy access of the developers, thus increasing the chances of developing a product that meets user requirements.

(Paulk, Curtis, Chrissis and Webster,1993, in Procaccino et al, 2002) suggest that one of the most important contributors to successful project development is end-user involvement. Amoako-Gyampah & White, 1997, in Procaccino et al, 2002) elaborate by emphasising that this involvement should occur in *all* phases of system development. Johnson (2001) would agree, advising that lack of user involvement has a strong affiliation to project failure. Giglio (1999) advises that involving end users early in the project greatly increases the chances of building something the users' will use and appreciate. Procaccino et al (2002) advise that an underlying assumption for project success is that projects will succeed if users/customers are highly involved and stay involved throughout the project.

A summary of important factors that relate to customer and/or user involvement in the software development process are given as:

- *User involvement is necessary in the development process* (Amoako-Gyampah & white, 1997, in Procaccino et al, 2002; Giglio, 1999; Procaccino et al, 2002, Paulk, Curtis, Chrissis and Webster,1993, in Procaccino et al 2002)
- *The user is knowledgeable and is authorised to make decisions* (Rakos, 1990).

Requirements

Saiedian and Dale (2000) highlight that requirements engineering (RE) is one of the most crucial steps in the software development process. According to Saiedian and Dale (2000) without a well written requirements specification, developers do not know what to build and users do not know what to expect. Saiedian and Dale (2000) continue, maintaining that without a well written requirements specification there can be no way of validating that the created system meets the users' needs. Rakos (1990) would agree maintaining that the requirements document states the user's problems

and the general solutions. Duvall (1995) explains that requirements are the conditions and capabilities needed to achieve the objectives of the system. (Procaccino and Verner, 2001; Pedhazur, 1982, in Procaccino et al, 2002) advise that a clear understanding of the problems that exist when establishing requirements is essential to the development of a successful system.

Rakos (1990) advises that time has to be spent with the user when producing this important document. Saiedian and Dale (2000) inform that the elicitation of requirements is a difficult process fraught with problems. Saiedian and Dale (2000) observe that poor communication and user resistance constitute areas of concern when gathering requirements. According to Saiedian and Dale (2000) user resistance is explained through a physical expression of opposition. User resistance is further identified by (Field, 1997, in Reel, 1999) as being one of the pitfalls of software development efforts. (Brown, Earl & McDermid, 1992) advise that a common problem with requirements elicitation is users who do not know what they want. The issue of communication is addressed by Brown et al (1992) who advise that communication problems can arise between the user and system analyst because there is no common understanding of the terms used by these two groups.

According to Duvall (1995) changes in requirements are one of the most prevalent problems with requirements. This would be corroborated by Johnson (?) who advises that by establishing a base level of requirements the effect of change will be reduced. Johnson (?) elaborates stating that by delivering minimal features users and sponsors can quickly see the results. In contrast to Johnson (?) Procaccino et al (2002) assume that a successful software project is that project where complete and accurate requirements are established from the start.

A summary of important factors that affect the RE process are given as:

- *Changes in requirements* (Duvall, 1995)
- *Involve the user in the RE process* (Rakos, 1990;
- *User resistance and communication* (Saiedian and Dale, 2000; Reel, 1999)
- *Establish basic requirements* (Johnson, 2001)
- *Complete and accurate requirements from the start*(Procaccino et al, 2002)

A summary of all the success factors identified by the literature analysis is presented in table 3.

Category	Success factor
Management	• *Well defined project objective* (Purba et al, 1995) • *Well defined project scope*(Purba et al, 1995; Field, 1997, in Reel, 1999) • *Minimise the scope of the project* (Johnson, 2001) • *Find a project sponsor /champion* (Johnson, 2001; The Standish Group, in Procaccino et al, 2002; Field, 1997, in Reel, 1999) • *Implement a post-mortem analysis* (Reel, 1999) • *Ensure effective communication between all parties* (Matheson and Tarjan ,1998; Purba et al, 1995) • *Incorporate effective software measurement into the process* (Paulish and Carleton, 1994; Capers Jones, 1995) • *Routines for effective quality control* –design reviews, code inspections-(Capers Jones, 1995)
Software development process	• *The use of a formal methodology* (Johnson, 2001 • *Use software tools* (Rakos, 1990; Capers Jones, 1995)
Estimation & scheduling	• *Include milestones* (Giglio, 1999; Capers Jones, 1995; Rakos, 1990) • *Set up realistic deadlines* (Field, 1997, in Reel, 1999; Glass, 1998b) • *monitor progress*(Capers Jones, 1995; Reel, 1999) • *Use estimating tools* (Capers Jones, 1995) • *Realistic estimations* (Johnson, 2001)
Development personnel	• *Build the right team* (Reel, 1999) • *Keep attrition low* (Reel, 1999). • *Allocation of roles to team members* (Rakos, 1990; Constantine, in Rettig, 1990) • *Skills of the development team* (Krishnam, 1998; Duvall, 1995; Field,1997, in Reel, 1999)) • *Give the development team ownership*(Giglio, 1999; Rettig, 1990)
The project manager	• *The ability to make good decisions* (Reel, 1999, Purba et al, 1995) • *Experienced project manager* (Johnson, 2001; Purba et al, 1995) • *Project managers have full authority to manage their project* (Procaccino et al, 2002)
Customers and users	• *User involvement is necessary in the development process* (Amoako-Gyampah & white, 1997, in Procaccino et al, 2002; Giglio, 1999; Procaccino et al, 2002, Paulk, Curtis, Chrissis and Webster,1993, in Procaccino et al 2002) • *The user is knowledgeable and is authorised to make decisions* (Rakos, 1990).
Requirements	• *Changes in requirements* (Duvall, 1995) • *Involve the user in the RE process* (Rakos, 1990; • *User resistance and communication* (Saiedian and Dale, 2000; Reel, 1999) • *Establish basic requirements* (Johnson, • *Complete and accurate requirements from the start*(Procaccino et al, 2002)

Table 3: summary of the identified success factors from the literature study

6.2 Presentation of interview material

In total five interviews have been conducted. The material generated from these sessions will be presented in the forthcoming section. The material will be presented in accordance with the categories: management, customers and users, requirements, estimation and scheduling, the project manager, the software development process and development team. In the context of interviews the term categories can be exchanged for themes which would better be in accordance with Patel and Davidson (1994).

As has been explained in section 5.3 an interview guide has been formulated. The interview guide may be of use when considering this material and can be found as supplement 3. The interview guide may be used to follow the topics which have been raised under each theme. The material outlined in this section is a segment of the original material. All excerpts in this section have been translated from their original Swedish by the author of this project. For a full transcription of the interview material the author of this project should be contacted.

In order to present the information in a form which favours the more general analysis presented in chapter 7 it has been decided that quotations will be used as often as possible to emphasise what actually has been said in the interview sessions. An analysis of this raw material is given after each topic. This is motivated by the fact that the author of this report feels that this approach better favours the continuity of the report. Additionally, the focus of chapter 7 can subsequently be altered to instead encompass a more general analysis which spans the whole report.

6.2.1 Management

In this sub-section material concerning the theme management will be presented.

Project objective

In total five questions were asked concerning the category management. In all cases uncertainty was encountered by all respondents concerning the term project objective. When this was exchanged for project goal the respondents were able to answer. When respondent 1 (R1) was asked what significance he attached to the projects goal with regard to a successful software project an acknowledgement for its relevance was acquired. This is apparent from the following excerpt:

> The goal of the project naturally has relevance. The project usually has a relatively clear goal. (R1)

Respondent 2 (R2) commented that delimitations were important and that such delimitations were usually defined. Respondent 3 (R3) laid a particular emphasis on this aspect, stressing its importance for the project:

> Its importance is decisive. Without a clear goal which is possible to verify then you have no way of knowing if you have completed the project or not. In other words, you have no way of knowing if you have a project or not. (R3)

R3 continues, emphasising further the importance of the project goal:

> Unclear goals have a tendency to collapse projects. (R3)

Respondent 4 (R4) emphasised the importance of roles when considering the objective of the project. According to R4 your position in the project influences a particular persons requirements/needs from the project objective:

> It is heavily role accentuated […] I believe that the goal should be adapted to fit both the person and the task that person is undertaking. (R4)

Respondent 5 (R5) explained that a well defined goal was essential for a project. A clear goal according to R5 was necessary in order to know exactly what to create:

> It is extremely important that one has a clear goal to steer towards. A clear goal is extremely important in order to know exactly what to create. (R5)

Analysis

The respondents were not familiar with the term objective. When this was exchanged for the term goal all respondents were able to answer. All respondents acknowledged that the goal of the project was a very relevant and important factor. Some respondents were more specific than others. For example, R3 maintained that the objective was decisive for the project. According to R3, if the project does not have a verifiable goal then one has no way of knowing if the project has been completed or not. R5, in line with R3, placed considerable emphasis on the importance of the project goal in the sense that without a clear goal one cannot create the requested product.

R4 offered an interesting perspective on the goal of the project by implying that the goal should be adapted to fit the individual and the task that individual is undertaking. A particularly strong sentiment is captured by R3 who maintains that unclear goals had a tendency to collapse projects.

Sponsor/champion

When asked for their opinions regarding a project sponsor three respondents shared a similar interpretation of this term whereas the fourth had a slightly different interpretation. R1 used the term internal customer. R1 emphasised the sponsor's role as bridging the gap between developer and customer.

> Yes, definitely. A project should include such a person, someone who we call an internal customer. As we usually have difficulties communicating with the customer we create a fictitious customer […] a person who both the development team and customer can liaise with. (R1)

R1 explains the consequences of a project lacking a sponsor:

> […] it becomes very difficult if you don't have such a person. The risk is great that the project will become very expensive. (R1)

R2 acknowledged the term sponsor and that it had, for several years, been present in a project management model used at R2s company. R2 considered a sponsor to be a valid inclusion in this model.R3 emphasised the economic role of the sponsor, laying emphasis on the fact that the sponsor provided the necessary funds needed to finance the project. R4 considered the relevance of the sponsor in relation to the size of the project. In either case R4 considered the sponsor to be a valuable asset:

> I believe it depends on the size of the project. In any case a sponsor is a very valuable person, someone who better sees continuity in the project. (R4)

R5 was of the opinion that sponsors contributed with important elements to the project:

> We have a steering group of which the sponsor is a member. The steering group is assigned the role of supporting the project manager in certain issues, for example resource issues. (R5)

Analysis

When asked for opinions regarding the relevance of a sponsor/champion in software projects 3 out of 5 respondents recognised this term in the context that has been identified in the literature analysis:

> *"(The Standish Group, in Procaccino et al, 2002) maintain that a project needs a committed sponsor or champion throughout. This sponsor should participate in the decision making process and encourage the same commitment from other stakeholders."* (p. 48)

The fourth respondent recognised the term sponsor in the economical sense, i.e. a financial provider. All five respondents, irrespective of interpretation, valued the presence of a sponsor.

The concept of the sponsor adopting a superior role is addressed by R1, R4 and R5. R1 explained that the sponsor assisted the project team with customer contact. R1 implied that the project team may experience difficulty with this aspect of project work and therefore the sponsor fulfilled a very necessary function. R5 explained that the term sponsor was used for a member of a steer group whose chief function was to support the project manager in certain issues.

Whereas R1 and R5 addressed specifics in the sponsor's role; R4 encapsulates the issue of superiority by considering the holisticity of the project. R4 maintains that the sponsor better sees continuity in the project which would imply that the sponsor adopts a position from which he/she can overlook the project.

Post-mortem analysis

When asked for their opinions concerning lessons learned from past mistakes all respondents were of the opinion that this was an important aspect sometimes overlooked in projects. R1 explains that too little time is dedicated to this aspect of the project. The fact that resources may not be available and the fact that this phase may not have been planned are given by R1 as plausible reasons for this deficiency:

> I believe that too little time is dedicated too this […] all the money is gone, no resources are left, one hasn't planned for analysis and feedback. (R1)

R1 points out however that this aspect is partially incorporated into the development process in use at R1s company:

> [..] this is partially built into the process we use. After each iterative stage we have an evaluation where everyone has the opportunity to give their thoughts about good and bad aspects of that particular stage. (R1)

R2 was of a similar opinion, placing emphasis on the fact that people are very tired at the end of the project:

> It is a deficiency in our projects and that is a shame. That plus the fact that when everything is done no-one has the energy to go through all that. (R2)

The response obtained from R3 was slightly different. R3 emphasises the importance of keeping this phase at the right level. R3 acknowledges the relevance of this phase but at the same time maintains that it should not be too long as no one would read it.

> It's important to keep it at the right level. It's a difficult balance. A report must be written but at the most 10 pages A4. People will read it then. (R3)

R4 considers the fact that when the project actually has been completed everyone involved in the project has at that stage been reallocated to a new project. R4 highlights that if a technical development project stretches over a long period of times to complete then no-one will remember what happened in the early stages of the project:

> […] it would be wonderful if it could be realised but when one has come so far that the project actually is complete and an evaluation is possible, everyone at that stage is gone and the project doesn't exist. […] besides, if it is a technical development project which stretches over a long period of time then no-one is going to remember what happened 2 years ago, 2 months ago or even 2 weeks ago as so much happens all the time. (R4)

R5 advises that if projects were not evaluated upon completion then mistakes and successes could not be capitalised upon in future projects:

> If we didn't do this then we wouldn't learn from mistakes or successes. By studying how the project has been run many things can be learnt. (R5)

Analysis

The concept of a post-mortem analysis was fervently acknowledged by all respondents. Both R1 and R2 maintained that this aspect was a regrettable deficiency in the projects these managers have led. R4 was of the opinion that such an analysis is usually hard to achieve. According to R4, when the project has reached that stage when it actually is complete project members have inevitably been reassigned. Furthermore, R4 informs that if the project has spanned a long period of time it is unlikely that people will remember events that occurred early on in the project. R5 offers a concise motivation for the contributions of a post-mortem analysis by ascertaining that if such an analysis was not conducted then lessons would not be learnt.

Communication

R2 maintains that communication between the stakeholders is a very important aspect to be considered in software projects. According to R2 this is also difficult to get right:

> Communication, we have noticed, is a very important, difficult aspect. To create a common vision of the goal (R2)

When asked how communications could be improved R2 answered that specifications were used during meetings and that the stakeholders spoke until a common vision was reached:

> Specifications. One has meetings where different groups within the project meet and go through these specifications. We speak until we are certain that we have a common vision. (R2)

R3 also values the presence of good communications. R3 observes that it is important that the customer is engaged from the beginning and is able to communicate his/her vision of the goal.

> Communication is important. It is important that the customer is engaged from the beginning, is interested and communicates his/her vision of the goal. If the customer can successfully do that then we have both a correctly defined goal and a close communication. (R3)

R3 continues explaining the problems which can arise if good communication is not established between stakeholders:

> […]there is a lesser risk that one falls into the-dare I ring and ask- trap. A dangerous trap to fall into because one chooses not to ring and ask about things they are unsure about. One makes a decision, the risk is great that the decision is wrong and the consequences are expensive. (R3)

R5 is of the opinion that good communication constitutes a key for project success:

> This is one of the keys to project success. Partly that good relations exist and that one has close contact and clear lines of communication. (R5)

Analysis

Three out of five respondents recognised the value of good communications. R5 even maintained that good communications constituted a key to project success. R2 advised that good communications are hard to achieve and that it was important to achieve a common vision for the project. By acknowledging the importance of a common vision R2 suggests a coupling back to the goal of the project or project objective. This affiliation would be endorsed by R3 who stresses the importance of an engaged customer and the ability of this customer to be able to effectively communicate his/her vision of the goal.

Software measurement

Software measurement is a term which encompasses aspects such as product size, lines of code, man hours etc. None of the respondents recognised this term in itself but recognised the individual components the term applied to.

When asked how much time should be dedicated to these aspects R1 explained that, depending on the projects size, two people working full time for 2-3 days was to be expected. R1 stressed the importance of establishing a cross section of viewpoints regarding these aspects:

> Depending on the size of the project one should dedicate 2-3 days, and that's two people working full time. After that we may have a 5-10 man forum where we don't go so deep. It's extremely important that one obtains varied points of view on this, extremely important. (R1)

When asked weather carelessness when estimating lines of code, man hours etc could affect the success of the project R1 maintained that it could. In R1s opinion this was because by carefully estimating such aspects and involving the people who ultimately will work on the project a common vision can be created. R1 maintains that much is earned by involving the team in these estimations. In addition to a common vision the team can establish which parts are difficult, which parts are easier, which parts should be completed first etc. the team also have a chance to discuss the product

R2 explains that the products main functions are established first then an estimate is made for how many man hours would be required to create these functions. R2 emphasises how important it is when giving estimates of time:

> extremely important. Often time is something which is underestimated. I have, on several occasions, witnessed that competence is required when estimating time. Especially when giving a price tag. (R2)

R3 observes that these aspects are influenced by the competence of the team:

> it is important that these aspects are considered […] I am project manager for a number of people and we are going to develop a software product. If this is their first project then these aspects are really important. If this is their 150 th project then they are going to require completely different guidance. (R3)

R4 recognised components such as lines of code, man hours etc. R4 informed that this information was used to estimate the project.

Analysis

The term software measurement was new to all respondents. The respondents did, however, recognise the individual components the term referred to: lines of code (LOC), man hours, scheduled time etc.

Varied points of view were gathered when discussing software measurement. Three of the five respondents acknowledged the importance of the individual components (LOC, man hours and scheduled time). R1 stressed the importance of establishing varied points of view with regard to these components whereas R3 advised that although aspects such as (LOC) were important there was a direct affiliation to the competence of the team. R3 means that if the project team is inexperienced then aspects such as LOC are important. If the team is experienced then such aspects are not so important. R2 focused primarily on the time components of software measurement: man hours, scheduled time etc. R2 considered these aspects to be extremely important and often underestimated.

Quality control

Routines for quality control include features such as: design reviews, code inspections testing etc. When asked if the same degree of thoroughness was required with these features as was required with software measurement R1 replied that this should be but was not the present case. R1 informed that design reviews and code inspections were implemented automatically. R1 cautions for over ambition when working with features for quality control.

> “design reviews, code inspections and such are usually implemented automatically. One should not be over ambitious with these things. It’s a fine balance but MOOSE (a development process) allows that quality checks can be omitted if the team is in agreement.” (R1)

R2 explains that his company have ambitions to be thorough with quality procedures. R2 informs that lack of time or money causes such things to be overlooked:

> “we are aware of these things, testing and such. We have ambitions to be thorough with these things but when time, perhaps even money is lacking such things go by the wayside. I believe that everyone is in agreement that these things, testing and such, are good things.” (R3)

R3 informs that quality procedures belong to the process. R3 considers such aspects to be directly attributable to the complexity of the product and the competence of the team.

> "These things belong to the process. They have to be there. Once again the total competence of the team is valid here. That and the complexity of the product." (R3)

R4 relates the issue of quality control to the size of the project and the lifespan of the product:

> "they are important, especially if the project has been going on a while. If the project is short and the product has a short lifespan then quality checks are not as important as something which is expected to last a while and will probably be updated." (R4)

R5 advises that routines for quality control have to exist before the product can be delivered.

> We must have routines before we deliver. We have routines both before and after delivery. (R5)

Analysis

When asked for their opinions on quality control and components such as: design reviews, code inspections etc, all respondents recognised the relevance of such features. There were, however, varying degrees of enthusiasm from the respondents when considering quality control. Both R1 and R3 advised that quality checks belonged to the process and therefore were implemented automatically. R3 was particularly enthusiastic about this part of the process maintaining that quality checks had to be there. R1, although acknowledging the importance of such checks, cautioned for over ambition maintaining that quality checks can be omitted if the team is in agreement. In these comments R1 would appear to suggest that one can be over thorough with quality control. This sentiment would be challenged by R5 who suggests that quality checks may be demanded by the customer.

R4 established a correlation between quality control and the lifespan of the product by informing that quality checks were vital for a product which is expected to last a while and may be updated. If the product was only expected to last a while then these checks were not as important.

6.2.2 The software development process

Information related to the theme the development process will now be presented.

Use of a methodology

All respondents recognised, to varying degrees, the value of a methodology when developing software. R1 considered a methodology to be affiliated to professionalism and necessary to guarantee aspects vital to the customer. R1 points out interestingly enough that the method does not guarantee the result:

> when you take on work you need to have a certain professionalism. You need, for example, to be able to promise that this and this will be delivered in 6 months and will cost X. You need to be able to guarantee certain things. Of course the methodology does not guarantee a result. (R1)

R2 as methods were relatively "heavy" to follow judgement should be used when considering their use. R2 thought that, in principle, methods should be used.

> in principle I believe so, of course one needs to use judgement as methods are quite heavy. A lot of time goes to following them exactly. One should use judgement depending on the size of the project. (R2)

R3s response was simple and to the point:

> They are very important. We rely on the method. (R3)

R4 explains that a method is useful if the same thing is to be repeated. When building a smaller product R4 feels that a method may not necessarily be required.

> If something is going to be reproduced then it is good to have a method. When developing a smaller product one can just get on with it. Make sure you write everything down because in a couple of months you won't know what you have. (R4)

R4 maintains however that methods have their place in larger projects:

> "[…]One can accomplish good things without a method, but when you come up to volume then you need a method otherwise you'll have chaos" (R4)

R5 is of the opinion that use of a methodology implies a common working practice. R5 informs that use of a methodology better enables the transfer of systems developers between projects.

> I believe that it is good to use a methodology because you create a common working practice […] it should be easier to move system developers between projects if one follows a methodology. (R5)

Analysis

All respondents were of the opinion that use of a methodology had its place in software development. Three of the five respondents were particularly enthusiastic about this aspect. R1 affiliated use of a methodology to professionalism and suggested that use of a methodology may guarantee aspects vital to the customer's peace of mind. R3 offered a particularly concise response by ascertaining that the everything rested on the method.

R5, in addition to R1 and R3, heavily endorsed the use of a methodology maintaining that a methodology enabled a common working practice. R2 informed that in principle a methodology could be used in software development but cautions that methodologies are time consuming if they are to be adhered to. R2 advises therefore that judgement should be used when considering the use of a methodology. R4 affiliated use of a methodology to project size informing that when one comes up to volume then a method will be required to avoid chaos

CASE tools

The respondents were not thoroughly familiar with the term CASE-tools. This term had to be explained. However, once this had been done the respondents were able to answer. When asked if these tools were necessary for the success of the project R1 maintained that the advantage with using the same tool was homogeneity:

> the problem is, if you take them away then everyone is going to use their own way. The advantage with everyone using the same notation is that it looks the same and it is quick. There are a lot of strengths in having the same tool. (R1)

R2 explained that his company have developed a development environment where automatic code is generated for certain functions. When asked if this tool was necessary to the success of software projects R2 felt that quality would suffer and cost would rise if this tool was not available.

> it would be more expensive and the quality would suffer. Standard functions, for example a table which utilises a user dialogue, can automatically be generated with code we can guarantee with the help of these tools. (R2)

Analysis

Two respondents gave favourable opinions regarding the use of CASE-tools. R2 emphasised aspects such as cost and quality, maintaining that such aspects would suffer if CASE-tools were not used. R1 explained that use of such tools implied certain strengths, i.e. speed in the process due to the fact that everyone was using a standard tool.

6.2.3 Estimation and scheduling

A presentation of the information concerning scheduling in software projects will now be given.

Milestones and regular reports

R1 considers realistic deadlines were important even if they were not always achieved. R1 informs that deadlines imply that resources can be gathered for the purpose of achieving a common goal:

> They are important irrespective of weather you meet them or not they have filled the function of gathering resources for the common goal. (R1)

R2 informs that in the beginning one may have a tendency to rush into things. Milestones, according to R2, provide stability in the sense that one can go back and see what needs to be completed before the next phase is commenced:

> At the start of the project one rushes in. It may be necessary to halt the process and go back to see what we said in the beginning. Then we see what needs to be completed before we go further. (R2)

R2 advised that it was important to get the team members to respect deadlines. R2 informs that it is important therefore that deadlines are realistic. R2 informs that the project can succeed if some deadlines are missed along the way but this is harder:

> Important! One must try to get the team members to respect the deadlines, which means that they must be realistic. The project can succeed if some deadlines are missed along the way but this will be harder. (R2)

R3 maintained that milestones or checkpoints were essential in order to confirm the status of the project with the customer. R3 advises that these checkpoints are relevant when progressing to the next stage of the project

> It’s these checkpoints where we agree that we can go further. We confirm the projects status with the sponsor and the customer. (R3)

R4 informs that milestones are important but that adhering to them may not always be realistic:

> […] these milestones are defined in the beginning and then they should be adhered to no matter what happens in the project. This is not always realistic. User requirements, or other prerequisites, may change. (R4)

R5 informs that milestones constitute the only way to clarify how something can be achieved and followed up within a specific timeframe.

> They are important. It's the only way to clarify what needs to be accomplished and followed up within a specific timeframe. (R5)

Analysis

When asked for opinions regarding the relevance of milestones in software projects all five respondents fervently acknowledged the relevance of milestoning and realistic deadlines.

R1 implies that deadlines fill the purpose of gathering resources for the purpose of achieving a common goal. R1 does not focus on meeting the deadline but instead emphasises the concentration of effort anchored to a specific timeframe. R5 encapsulated the essence of milestoning by ascertaining that this was the only way by which to clarify what needs to be done within a certain timeframe

R3 enforced the importance of monitoring the project. Milestones or checkpoints according to R3 constitute those points in time when the project progresses from one stage to the next. The status is confirmed with the customer and progression to the next stage is authorised. Confirming the status with the customer would imply that the process must have been monitored in order to ensure that all phases encompassed by the milestone have been achieved.

R2 offers further support to the relevance of milestones and the monitoring of progress. R2 explains that at the start of the project the tendency may be to rush in. There may come a point where the process has to be stopped so an evaluation can take place to see what needs to be completed before the next phase commences. R2 advises that missing deadlines may, but not necessarily, compromise the success of the project. R2 further identifies the importance of realistic deadlines by ascertaining that it is essential for team members to respect the deadlines that have been established. According to R2 this can only be established if the deadlines are realistic.

R4 explains that deadlines are established at project start and should consequently be adhered to. According to R4 this is unrealistic as user requirements or other prerequisites may change. Predicting the occurrence of such events is not easy but by allowing for them then perhaps more realistic deadlines could be generated.

A further reference to scheduled time can be found in the responses obtained when the respondents were asked for their personal opinions on those factors that constituted software project success. Three respondents maintained that delivery within estimated time constituted a success factor.

No specific questions were asked related to software project estimation. However, when asked for personal opinions as to what constituted software project success three out of five respondents ascertained that delivery within budget was a factor for success. This can only be interpreted as implying that realistic estimation is extremely important

6.2.4 Development personnel

Information related to the development process will now be presented

Individual competence of the team members

When asked for opinions concerning the importance of team members R1 gave a decisive answer:

> I believe it is very important but I believe that the work climate and job satisfaction are more important.(R1)

R2 was keen to point out that this was very much dependant on the type of competence in question.

> […] it depends what competence we mean here. In some technological areas there are many people to choose between. In other, smaller, areas everything hangs on 2-3 people. Naturally it's important to get those people into the project. (R2)

R3 incorporates a similar line of reasoning to R2s. R3 adds that it is important with a breadth of competences in the team. R3 highlights that a passion for the task may even be more valuable.

> It's really important to have a breadth of competence in the development team. It's no use having 10 people who are great at assembler programming. On the other hand I might be interested in one. Often it's more important with an interest for the task than the necessary knowledge. To have passion for what we are going to build.(R3)

R4 informs that technical competence is insufficient:

> If we set up a team for a 2 week job someone might say: this person can develop software, he/she can write code. It's not enough. If we are to do something in such a short space of time then it needs to be exactly the right person […] it has to do with how one is as a person so there are several types of competence. (R4)

R5 considers competence to be a prerequisite for working in project form. R5 advises that when a project team is put together the intention is to acquire the people who have the best skills for the task in hand:

> One of the reasons we work in project form is that when we receive a job we try and put together a team from different departments and units. Naturally we try and acquire those with the best skills for that type of project, under the assumption that they are available. (R5)

Analysis

When considering the competence of individual team member's three respondents claimed that identifying individuals with specific skills was important for the project. R2 explains that in some technological areas a multitude of people can be chosen between. In other areas everything hangs on 2-3 people. R3 means that it is important to involve such people. R5 would definitely agree by firmly establishing that one obviously tries to acquire the people with the best skills for the task at hand.

R3 informs that building a team where 10 people are competent in, for example, a specific programming language is not desirable. R3 explains that one person with that particular competence is, on the other hand, very desirable.

R4 is keen to inform that technical competence is insufficient when working in projects which do not last long. R4 maintains that when a project is put together for a two week job then one needs exactly the right person. R4 would imply that the personal qualities of a team member may be more important in this context than his/her ability to perform a specific task.

Build the right team from the beginning

R1 was uncertain as to whether it was necessary to build the right development team from the beginning. When asked, however, if the development team could be built successively throughout the course of the project R1 felt that to a degree this was possible.

> I think this is possible to a degree. For example you might want to add a verifications specialist or a protocol specialist at later stages in the project. (R1)

R1 indicates, however that moving people from projects may be inadvisable:

> I believe that people should not be moved in projects if the move cannot be motivated. You should not move people for economical reasons. (R1)

R2 felt that it was certainly an advantage to have the correct development team from the beginning of the project

> Of course it is better when everyone has the history from the beginning. Its easier that way. (R2)

R2 continues, warning that when new members are added information inevitably gets lost when bringing that person up to speed:

> When a new member arrives one obviously shares all the available information but something always gets lost in this transfer. (R2)

R3 is of a similar opinion to R2:

> If you are successful building the right team from the beginning then that is an advantage. (R3)

The social aspects of team building are addressed by R3 who observes that this can inevitably affect the output of the team if this process needs to be repeated for new members:

> In the beginning everyone gets to know each other, everything is positive and it feels good […] in the situation of a new person coming in the induction phase starts over[…] if you keep adding people you become fully occupied with the social aspects and you end up with no output. (R3)

According to R4 in an ideal situation the right team can be built in the beginning but this is not often the case. R4 advises that team members must inevitably apply team members successively throughout the course of the project

> You are forced to add members successively. That's the way it is. The dream is that you can get the people with the right competence from the start and that everyone gets on and no one leaves. It doesn't happen. (R4)

R5 informs that building the right team from the beginning may be difficult as the ultimate goal and scope of the project may well not be fully defined. R5 suggests that events occur during the project which results in personnel being added to the team.

> At the start of the project the scope and the goal may not be well defined. Things happen during a project which results in people being added. (R5)

Analysis

Two of the respondents considered the building of the right software team at the start of the project as being advantageous. R3 explains that at the start of the project a certain social decorum takes place where everyone gets to know one another. If project members keep being added then this decorum needs to be repeated and as a consequence output is minimised.

In contrast, two of the five respondents maintained that building the right team could only be accomplished by successively adding team members throughout the course of the project. R5 motivates this by explaining that at the start of the project the project goal may not be well defined. As the goal becomes better defined the addition of more people may be required.

Attrition

People leaving the project does not necessarily imply negativity. R1 explained that this seemingly unattractive event could have advantages:

> It can be healthy for a project to adopt new people. Situations can arise where something may not be as good as it could be. One can then choose to exchange one of the people and try to make something good out of it. (R1)

R2 is of the opinion that people leaving the project is undesirable but acknowledges that positive aspects can be drawn from this situation.

> It's a disadvantage to lose an asset. I've experienced it and it was annoying. […] it can also have benefits to get someone new, someone who might question what we take for granted. (R2)

Analysis

Attrition in itself is a negative feature. Indeed, as R2 explains, losing an asset is undesirable. However, both R1 and R2 advise that upon losing a person the project may acquire a new person. This new person may then be able to assess the problem situation from a different perspective and thus bring to light issues which may have, up till then, been invisible for the other team members.

Allocation of roles

When asked if the allocation of roles to team members could improve the motivation of the team R1 replied that it was important have an area of responsibility. Furthermore, R1 emphasises that this allocation of responsibility is important for the project to be able to progress beyond a certain stage:

> You should at least have an area of responsibility. […] is not always easy to come to a consensus in software development. There are many ways to solve a problem. You have to have a role which gives a person the right to say : this isn't the best solution but we'll take it because we have to go further. (R1)

R2 maintained that it was more important to establish what the individual should accomplish. However, R2 agreed that a clear description of roles was an important base:

> More importantly everyone should know what they should accomplish […] as a base it is important to have clear role allocation. (R2)

R3 makes a correlation between role allocations and goals.

> When it's clear what you have to do you do it. We're back to goals but goals on a lower level, that my task is to complete this within this time. That's my responsibility. (R3)

R4 suggests that role allocation is important to establish certainty about what needs to be accomplished:

> […] if people don't know what their role or task is then they don't know what they should be doing and everything is uncertain. (R4)

R4 advises that the nature of project work often implies that people need to change roles. According to R4 this can only be effectively accomplished if people know exactly what they should be doing:

> […] in a rapidly changing project roles can change. You need to be very thorough in making sure that people know what they should do. (R4)

R5 informs that some form of role allocation is important but this shouldn't be exaggerated as project members are often required to take on different roles:

> On the one hand I believe that some form of role allocation is important but on the other hand I feel that perhaps it's a little exaggerated because in a project environment a team member is required to take on different roles. (R5)

Analysis

The concept of role allocation was acknowledged by four out of five respondents. However, varying degrees of importance were attached to this concept. Both R4 and R5 advised that roles can change in a project and that one needs to be aware of this. R4 and R5 accept, however, that roles do have their place. R1 explains that an area of responsibility should, at the very least, be allocated. The establishment of areas of responsibility would be endorsed by R2 who advises that it is important for the team member to know what to accomplish

Ownership

Giving the team members influence over that they are working with is considered by R1 to be very important:

> That is very important. When the project starts it may be that not everyone agrees with the project plan […] we discuss it and arrive at a decision where everyone has had input. (R1)

R2 advises that freedom can be allocated but a boundary needs to be established for the actual design so that team members do not stray outside the limits for that boundary:

> It is important to establish a boundary and then allocate freedom within that boundary. (R2)

R4 maintains that the concept of ownership is different from individual to individual. Some team members want to have influence, others are not so interested.

> For some people it is very necessary, that they have influence. Others don't care less. This is something that needs to be considered from project team to project team. (R4)

R5 endorsed ownership:

> I believe it is important that they feel free, that they are allowed to be creative and make suggestions. I think that's important. (R5)

Analysis

Four respondents considered the concept of ownership to be important in software projects. R1 and R5 were particularly enthusiastic about this concept. R1 advised that it was important that the team member feels he/she has input. R5 would agree, ascertaining that the freedom to be creative and contribute with suggestions was an important aspect of project work. R2 and R4 were restrictive in their comments concerning ownership. R2, in line with R1 and R5, did acknowledge the relevance of ownership but maintained that it was important to establish a boundary within which creative freedom could be granted. R4 explained that ownership needed to be considered from project to project. According to R4 some people want to influence the design, others do not

6.2.5 The project manager

A presentation of the information concerning the project manager in software projects will now be given

Qualities of the project manager

When asked what qualities were essential for a project manager R1 laid particular emphasis on the project manager's personal qualities:

> He should be communicative, seen and heard and relatively technical. Most important is that he is available and listens. (R1)

R1 was not of the opinion that experience was not necessarily a prerequisite for managing projects. R1 advised that experience could even be a disadvantage.

> […] experience can be a disadvantage, one makes the same mistakes etc. I believe that experience is good but not completely necessary. (R1)

R1 highlights that the project manger must be prepared to take responsibility, even when the project does not go as planned. The project manager should also encourage a positive working environment:

> It is very important to take responsibility […] when the project doesn't go as planned the project manger has to explain for the executives why this is the case. The project manager drives the project. he should ensure a positive working environment. (R1)

R2 advised that a project manager should be able to manage multiple tasks and display good judgement:

> I believe that it's good to be able to manage multiple tasks, be concise with what is expected and identify discrepancies. (R2)

When asked if a project manager should be experienced R2 maintained that this may be necessary if the project was large.

R3 regarded the project manger as the driving force for the project:

> The ability to create a positive atmosphere so that everyone is passionate for the task in hand. (R3)

R3 considers experience to be an important quality when managing software projects:

> […] It's very important […] software projects are typical examples of problematic situations, there are many pitfalls […] you need that experience. To manage software projects you must have worked with software. (R3)

In R3s opinion the ability to make a decision was one of the project manager's most important tasks:

> Decision making is one of the project managers most important tasks […] he has to make a decision, he can't hide behind anyone. (R3)

In R4s opinion a project manager must be prepared to take the blame for anything that may go wrong in the project. R4 highlights that it is always the project manager's fault:

> The project manager must be prepared to bear responsibility for everything. It is always the project manager's fault, never the individual. (R4)

R4, in line with R2, maintained that experience was more relevant when managing large projects. R4 considered personal qualities to be more important. R4 was of the opinion that decision making was a very important quality for a project manager. Additionally, R4 maintained that the project manger should have the ability to make the decision in that moment where the decision is required:

> Very important. You can't stand there and ponder over something or say that you'll get back to someone. It doesn't work. (R4)

R5 observes that a project manager needs to have leadership qualities. He needs to be able to handle both the user and his development team. R5 informs that experience is a good quality but there will always be a first project. R5 maintains that the capacity to make decisions is a very desirable quality even if a bad decision is made.

> Even if one makes the wrong decision, a decision needs to be made. (R5)

Analysis

When asked for their opinions on what constituted a good project manager qualities such as communicative, inspirative and prepared to take responsibility were forthcoming. The inspirative qualities of PMs were specifically addressed by two respondents, R1 and R3, who maintained that a PM should ensure a positive working environment.

Three respondents recognised experience as a valid quality for project managers although only R3 maintained that experience was an essential quality for a project manager

All respondents were convinced that the ability to make decisions was an essential quality for a project manager. This quality is captured effectively by R3 who states that the project manger has to make a decision; he can not hide behind anyone.

The project manager is authorised to run his/her project

R1 considers that the project manager should have authority to manage the project within his/her area of responsibility.

R2 was convinced that the project manager should be granted authority to manage his/her project

R4 maintains that the project manager should have authority to manage his/her project and make any type of decision. If this is not possible then effective channels of communication should exist between the project manager and that person who does have that authority:

> Yes, or effective channels exist between the project manager and those that have that authority. It is more important that the project manager is authorised as running a project implies that many decisions need to be made. (R4)

R5 informs that because so many projects run at the same time giving project managers full authority to run their projects is not desirable. R5 maintains that some kind of central coordination unit is required:

> On many occasions several projects run at the same time. Each project manager cannot be given full authority to run his project. There must be some kind of central coordination. (R5)

Analysis

Three respondents were of the staunch opinion that the project manager should be granted authority to manage his/her own project. R5 felt, however, that this was not possible due to the fact that many projects ran at the same time. R5 maintained that some kind of central control was necessary in order to supervise all these projects that run at the same time

6.2.6 Customers and users

The information related to the theme customers and users will now be presented.

User involvement in the whole development process

All respondents acknowledged the importance of user involvement in the development process. There were, however, varying opinions with regard to the extent of this involvement. All respondents considered customer and user to be one and the same.

When R1 was asked weather the user should be involved in the whole development process a correlation was made to specific type of project:

> " I think that is good when a human computer interaction (HCI) project is being run. If we are building something the customer will never see then I don't see any value in that." (R1)

R2 had a different opinion on the extent and validity of user involvement in the development process:

> " I am convinced. Especially in the beginning so we can agree on what should be built. After that we can check periodically with the customer." (R2)

when asked if the user can be omitted from certain phases of the process R3 pointed out that this often was the case. According to R3 there are phases in the project where user involvement actually becomes less attractive.

> "the customer can be omitted from certain phases[…] somewhere there is a boundary where it no longer is desirable to involve the customer because this will just cause problems. For the most part this is due to the fact that the customer lacks the necessary technical knowledge." (R3)

R3 does, however, acknowledge user involvement as being relevant:

> "the customer is definitely important. Take the process for example: how do you work? Do you really follow this process? The customer needs to be there and check this." (R3)

R4 would agree with R3s first comments that the user can be omitted from certain phases. R4 acknowledges, however, that continuous not constant user involvement throughout the development process is necessary.

> […] there is a situation in the beginning when it is important to involve the user. After that we enter a phase where we try to "develop something" and may just require the user's involvement now and then. It is not profitable to involve the user constantly during the process. On the other hand the user should be involved continuously. (R4)

R5 is of the opinion that the user should be involved from beginning to end but not necessarily in every phase:

> Not the whole process because we engage ourselves in things the user won't understand; technical stuff and so forth. The user should naturally be involved from beginning to end but perhaps not in all phases. (R5)

Analysis

All respondents maintained that the user should be involved in the development process. The chief tone of all responses was that the user need not explicitly be involved in every phase as this may cause complications due to the user's lack of technical knowledge.

The user is knowledgeable and is authorised to make decisions

R2 informs that someone on the customer side has to be authorised to make decisions. However, R2 is of the opinion that if the supplier is familiar with the environment where the system is to be implemented then this may not be such a major problem:

> Someone on the customer side, preferably a knowledgeable user, must be authorised to make a decision. Once again, if the supplier is familiar with the user's workplace then this may not be a problem. (R2)

R3 was of the opinion that such a user was an essential requirement for a project. R3 uses the term active customer:

> Another important point is that the project has an active customer. An active customer campaigns for a successful project. (R3)

When asked whether an active customer/user should have certain competences R3 felt that the only competence required was the ability to explain advantages and disadvantages with different solutions:

> […] it is a competence to be able to explain and point out advantages and disadvantages with different solutions. (R3)

R4 informs that if the user does not have this authority then effective channels of communication should be established between the user and that person who has this authority:

> If the user doesn't have this authority then effective channels are required between the user and that person who has this authority in order to avoid unnecessary waiting time. (R4)

R4 explains the consequences for the project if this aspect is neglected:

> […] the project plan and project goal can suffer severely. (R4)

R5 claims that a user who is authorised to make decisions is an important asset for a project. According to R5 the project loses time if the user is not authorised to make decisions:

> It is extremely important to have users tied to the project who are authorised to make decisions […] you lose time if you have users that are not empowered to make decisions. (R5)

Analysis

Four of the respondents maintained vigorously that a user who was authorised to make decisions was a very valuable asset to a project. The consequences for a project that lacks a user authorised to make decisions are neatly encapsulated by R4 who advises that the project plan and project goal can suffer severely if such a user is not involved in the development process.

6.2.7 Requirements

A presentation of the material concerning requirements in software projects will now be given.

Involvement of the user in the requirements elicitation (RE) process

R1 offers a concise motivation for the involvement of users in the RE process:

> I believe that the user should be present when establishing requirements. It is these requirements which are interesting. (R1)

R2 informs that the level of interest, engagement and the ability to provide opinions varies from user to user. R2 emphasises that one should always try and find an interested used that is willing to help.

> […] it depends on the user: their level of interest, engagement and points of view. you should always try to find an interested user who can help and provide opinions. (R2)

R3s response was in line with R1s:

> It is the user who provides the real requirements. (R3)

R4 acknowledges that user involvement is very positive but due to the fact that the user may not be able to express the requirements this involvement is not decisive for project success:

> In most cases the user has a feeling for what he/she wants […] the presumptions for success are greater if the user is involved but not decisive. (R4)

R5 considers user involvement in the RE process to be somewhat of a key to the whole process. R5 informs that if the user is not involved in this phase then much will be missed:

> We may miss a lot. We have to get the user to talk about what he wants and how it is supposed to look. It's one of the keys to the whole process. (R5)

Analysis

All respondents were enthusiastic about involving the user in the requirements process. R2 does, however, touch upon the issue that the user has to be motivated for this task.

Changes in requirements

R1 explained that by using use-cases (a specific notation associated with the Unified Modelling Language- UML) increases traceability when managing changes to requirements:

> We try to explain requirements with use-cases […] you have already done an analysis of which components are affected by that use case. You can trace back and see what needs to be changed. (R1)

R2 maintained that changes were managed by saving old requirements documents. One could then see how these documents change over time:

> […] save old versions to see how they have changed over time. (R2)

R3 explains that requirements should be firmly anchored. If changes are to be made the person that makes that decision should have a holistic impression of that he/she is going to change:

> Ensure that the requirements are anchored […] this is important. It may occur to someone that this requirement cannot be realised, we'll do this instead. The person who makes that decision may not have the complete picture in front of him/her. (R3)

R3 warns of the consequences for the project if changes to requirements are not managed effectively:

> You verify the requirements with the customer. If you change something you get another product. When you verify later with the customer and discover that changes have been made it can be very expensive compared to what it would have been if you had managed the changes correctly. (R3)

R4 enforces the necessity for a process when changes to requirements are made. Changes cannot be conferred verbally or changed continuously:

> You should have some kind of process for managing changes to requirements. You can't have a situation where someone says change that 4 for a 2. You can neither have a situation where requirements are changed continuously. We must have a process which dictates that we have a review and then we can change the requirements. (R4)

R4 explains the outcome if requirements are not managed effectively:

> It's not certain that you achieve what you intended. It may well be that you all of a sudden don't know which requirements are valid. (R4)

Analysis

Four out of five respondents were of the opinion that a process or procedure was necessary for managing changes in requirements. The explanations concerning this were varied and aspects such as verification and traceability were touched upon. The aspect of verification is clearly addressed by R3 who maintains that if changes are not managed effectively then the consequences can be expensive

Establish complete or basic requirements from the beginning

When asked if one should establish complete, detailed requirements or simply basic requirements from the beginning R1 advised that it was a real advantage to work with requirements on a higher level.

> You should work on a higher level. You shouldn't look at the small details […] we want to have requirements structured on a high level, preferably as use cases. I believe it's a real advantage if we can receive requirements structured on a higher level. (R1)

R2 maintained that it was important to identify all the main functions in the beginning. According to R2 the level of detail can be evaluated in accordance with its associated function:

> I believe that all main functions should be established at the start […] one needs to evaluate exactly what we need in order to build a function. Those things we can wait with we don't ask about. (R2)

R3 explains that the level of detail concerning requirements is dependent on the task in hand:

> If it's a big project its dependant upon where you are in the project. the level of detail concerning requirements is much higher when we come down to those requirements the constructor needs. (R3)

According to R4 there is no one way which is better, or more desirable than the other:

> You could say that one leads to the other. If you receive detailed requirements then you can always work backwards and establish the fundamental requirements or vice versa. (R4)

R5 acknowledges that it is necessary to identify the fundamental functions in order to establish the scope of the project. R5 informs that details need to delved into almost at once in order not to miss anything.

> One needs to get the fundamental requirements in order to understand the scope […] pretty much straight away you need to delve into details. You need to do that in order not to miss anything. (R5)

Analysis

When questioned on their opinions concerning the level of detail of requirements three respondents acknowledged that it was important to establish a higher level of requirements in the beginning. R1was of the opinion that establishing deeper levels of detail was unnecessary when working with software. R2 and R5 would seem to agree

stating that it is important to identify the main functions at the start of the project. R2 effectively justifies why details can be avoided by explaining that if you do not need it do not ask about it.

Whereas R2 and R1 tended to dismiss the necessity of deeper levels of detail in requirements R5 advised that once the fundamental requirements had been established it was necessary to delve into details. According to R5 this was necessary in order not to miss anything

User resistance

R2 maintains that each user is different. Some users know what they want early in the project whereas others do not realise this until after they have used the system. R2 informs that with users who know what they want the chance of building the wrong functions are minimal. R2 is of the opinion that it is harder to achieve a successful project without a committed user:

> Users are different: some know what they want early on in the project, others come to life after they have used the system a while. With the first sort the risk of building the wrong functions is minimised […] usually it's harder to achieve a successful project without a committed user. (R2)

R2 observes, however, that a committed, unknowledgeable user can be troublesome. R2 advises that if the system's supplier is familiar with the environment where the system is to be implemented then it may be cheaper and more effective not to anchor everything with the user.

R4 maintains that if the user does not collaborate then the project has to draw its own conclusions. R4 informs that this may not necessarily be negative as the user may not have a clear vision of what they want.

> If the user doesn't collaborate and share his/her opinions then the project is forces to draw its own conclusions, what needs to done, what the requirements are. This isn't always negative, especially if the customer doesn't know what he/she wants.

R4 suggests a solution for lack of user collaboration. R4 advises that the project can itself establish the requirements. These can then be confirmed with the customer for review and approval.

> […] one way to solve this is for the project to establish the requirements etc. these can then be confirmed with the customer for review and approval. (R4)

R5 was of the opinion that if the user is unable to dedicate sufficient time to the project then the success of the project will be threatened:

> If the user cannot dedicate that time which is necessary then the success of the project is definitely affected. (R5).

Analysis

The concept of user resistance was recognised by two of the respondents as being potentially detrimental to project success. R2 informs quite categorically that it usually is harder to achieve a successful project without a committed user:

R5 was of the opinion that if the user did not dedicate enough time to the project then the success of the project will be threatened. R4 explained that if the user was not committed then the project would be forced to make its own conclusions. According

to R4 this was not necessarily disadvantageous as the user may not know what he/she wants.

6.2.8 Respondents opinions on what characterises a successful software project

A presentation of the material relating to the respondents own opinions regarding software project success will now be given.

When asked for opinions regarding software project success R1 maintained that time and cost were the most important variables to consider:

> I am orientated towards the result. I think that if you have delivered within the allocated time and within budget then it's a success. It's that simple. (R1)

R2 would agree, also considering time and cost to be the decisive variables pertaining to project success.

> Deliver in time and within the estimated price. (R2)

R3 did not consider time and cost to be the most important aspects to consider. According to R3 customer satisfaction was the most important aspect to consider:

> […] a project can double its costs and go behind schedule but if the customer is satisfied then you've got a successful project. (R3)

R4 informs that if the product is at all is used is an indicator for success. Furthermore, the product should be used for that for which it was intended and gives the customer the requested functionality.

> The product should be: used, used for that which it was intended and gives the customer the requested functionality. (R4)

From the perspective of the project R4 ascertains that project success can be derived if it is possible to follow up if the goal has been reached, how the project's economy has stood and if experiences have been recorded before the next project begins.

R5 informs that delivering what the customer has requested constitutes success. Additionally, and in accordance with R1 and R2, R5 considers time and budget to be relevant variables when considering project success:

> To be able to deliver what the customer has requested within the timeframe and within the budget. (R5)

Analysis

Three out of five respondents staunchly maintained that delivery within time and budget constituted factors for software project success.

Customer satisfaction was a concept addressed by three of the respondents. R5 claims that delivering what the customer requested constitutes a success factor. R4 would agree, adding that the product should provide the customer with the requested functionality. R3 would appear to regard customer satisfaction as ultimately decisive for software project success.

6.2.9 Summary

A presentation has now been given for all material gathered throughout the course of this report. The first stages of the analytical process have also been presented together with the raw interview data. The author of this report feels that this approach lends

itself towards better continuity in the report and allows chapter 7 to better focus on a more specific analysis that covers the whole report.

7 Analysis and conclusions

The purpose of this section is to present a thorough analysis of the material gathered and presented in chapter 6. The analytical process has already been commenced in the previous chapter thereby allowing the focus of this chapter to shift to a more specific, anchored analysis that will propose to satisfy the aim of the report which has been given as:

Does the framework show potential for application in software projects where the purpose is not to build an organisational information system?

Before the analysis begins a cross check can be initiated to see if any of the report's objectives have been fulfilled. The following quotation outlines the purpose of the literature analysis in the context of this report:

> *"By studying documented case studies of successful software projects a deeper understanding of the factors concerning software project success can be realised."* (p.31)

A full account for how the literature analysis was conducted has been given in section 5.1. What can be added here is that the literature analysis has functioned extremely well for the purpose of identifying a multitude of success factors specific to the type of studied software

These success factors have been summarised in table 3, p. 54 but are shown again for analytical purposes in sub-section 7.1.1. The literature survey has provided a firm base from which an interview survey has been founded. In order to conduct the interviews suitable companies had to be identified.

On the basis of this account it can be ascertained that the first three objectives, given in section 3.1, have been fulfilled.

The analysis will be presented in two sections. In section 7.1 a connection to the framework identified and discussed in section 2.4 is re-established. This section will attempt to investigate if the framework has potential for use in non IS software projects by corroborating the success factors originally identified by Aggestam (2001) found in table 1, p. 19. Section 7.2 will further attempt to satisfy the aim by addressing the factors that are not encompassed by the framework and investigating if these factors can, if at all, be incorporated into the framework. The chapter will conclude with a final analysis which will outline the main findings of this report and will ultimately provide a point of reference for chapter 8.

7.1 Analysis of the material in the context of: stakeholder, objective and remaining

It has been firmly established in section 2.4 that the framework as proposed by Aggestam (2001) is founded on success factors that have subsequently been divided into three categories: stakeholder, objective and remaining. In order to satisfy the aim of the report and find out if the framework is indeed applicable in software projects where the purpose is not to build an organisational IS then a similar categorisation must be attempted using the material gathered and presented in chapter 6. Figure 13 graphically portrays this intention.

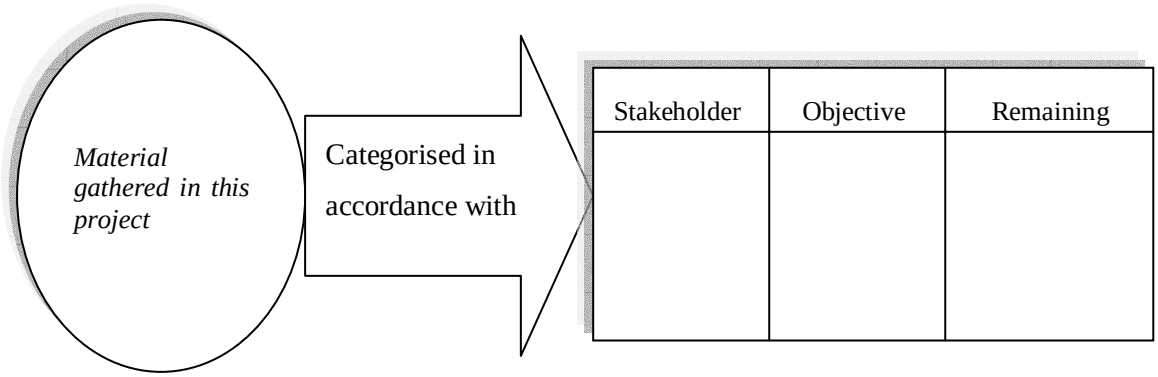

Figure 13: diagram showing the intention to categorise the material identified in this report in accordance with the categories: stakeholder, objective, remaining.

7.1.1 Substantiating the framework

Table 4 shows how the material identified during the course of this particular report has been aligned in accordance with the category stakeholder. The additional categories: objective and remaining have been accounted for in tables 5 and 6. The success factors to the left are those factors originally identified by Aggestam (2001). The information to the right are those factors or excerpts from the interview material which corroborate or support those factors identified by Aggestam (2001).

Using table 1 on page 19 as a reference, one can see that the category stakeholders is particularly well represented and all success factors identified by Aggestam (2001) can be substantiated by material gathered during this report. For the purpose of clarity it should be mentioned that the text in italics *effective communication*, located under the category stakeholder is associated with the following success factor identified by Aggestam (2001)

- *To involve the right stakeholder and manage their confidence and knowledge needs*

The attributory factor *effective communication* has been affiliated to this success factor as a means for fulfilling this success factor. Additional means are given by Aggestam (2001) as being: *human cognitive constraints, training and support*. These three additional attributory factors are missing from table 4 as they can not be substantiated by the findings of this report.

Stakeholder	
To achieve high user satisfaction	[…] a project can double its costs and go behind schedule but if the customer is satisfied then you've got a successful project. (R3) […] to be able to deliver what the customer has requested […] (R5) the product should be […] used for that which it is intended and gives the customer the requested functionality. (R4) *Build the right team (*Reel, 1999)
To involve the right stakeholder and manage their confidence and knowledge needs.	*User involvement is necessary in the development process* (Amoako-Gyampah & white, 1997, in Procaccino et al, 2002; Giglio, 1999; Procaccino et al, 2002, Paulk, Curtis, Chrissis and Webster,1993, in Procaccino et al 2002) "the customer is definitely important. Take the process for example: how do you work? Do you really follow this process? The customer needs to be there and check this." (R3) *Involve the user in the RE process* (Rakos, 1990; I believe that the user should be present when establishing requirements. It is these requirements which are interesting. (R1) *The user is knowledgeable and is authorised to make decisions* (Rakos, 1990). Someone on the customer side, preferably a knowledgeable user, must be authorised to make a decision […] (R2) It is extremely important to have users tied to the project who are authorised to make decisions […] you lose time if you have users that are not empowered to make decisions. (R5) *Experienced project manager* (Johnson, 2001; Purba et al, 1995) […]To manage software projects you must have worked with software. (R3)
Effective communication	*Ensure effective communication between all parties (Matheson and Tarjan ,1998; Purba et al, 1995)* This is one of the keys to project success. Partly that good relations exist and that one has close contact and clear lines of communication. (R5)
To create a positive attitude	[…] The project manager drives the project. He should ensure a positive working environment. (R1) the ability to to create a positive atmosphere so that everyone is passionate for the task in hand. (R3)
Championship, committed sponsor	*Find a project sponsor /champion (Johnson, 2001; The Standish Group, in Procaccino et al, 2002; Field, 1997, in Reel, 1999*) It becomes very difficult if you don't have a sponsor. The risk is great that the project becomes very expensive. (R1)

Table 4: categorisation of success factors in accordance with the category stakeholder

All success factors identified by Aggestam (2001) can be directly aligned to similar success factors identified in this report. This process of alignment will continue with the category objective.

objective	
To define the objective *To define it in different frames and at different levels of detail*	*Well defined project objective* (Purba et al, 1995) Its importance is decisive. Without a clear goal which is possible to verify then you have no way of knowing if you have completed the project or not. In other words, you have no way of knowing if you have a project or not. (R3)
Accepted objectives Good selection and justification	It is extremely important that one has a clear goal to steer towards. A clear goal is extremely important in order to know exactly what to create (R5) I believe that the goal should be adapted to fit both the person and the task that person is undertaking (R4)

Table 5: categorisation of success factors in accordance with the category objective

The category objective is also well represented with 3 out of the original 4 success factors being substantiated. The success factor which can not be supported by material identified in this report is:

- *Meet business objective*

There is, however, a plausible explanation for this. McDermid (1990) advises that the difference between developing organisational ISs and, for example, an embedded software product is the domain of the problem to be addressed. This has been discussed in section 2.1 where McDermid (1990) advised that the parameters governing the embedded system could easily be identified as these parameters were often of a physical, quantifiable nature.

This stands in contrast to an IS where according to the definition given in section 2.1 an IS consists of: hardware, software, data and people. This would imply that the parameters concerning an IS are more varied, and perhaps not as quantifiable as those concerning the embedded system. This would be endorsed by Andersen (1994) who informs that constructing an IS is considerably more than a technical undertaking. More specifically, the IS should support the organisation it is implemented in (Andersen, 1994; Avison & Shah, 1997). In order to support the organisation the IS is consequently required to enhance the organisations ability to run as a business. This concept is addressed in sub-section 2.3 where characteristics of IS and software success were discussed. The ISs ability to support the business function of the organisation is specifically addressed by Wateridge (1998) who informs that one of the criteria for IS success is that the IS achieves its business purpose in three ways (strategically, tactically and operationally).

So far, factors identified in this report have successfully been aligned to success factors in the framework's two critical categories: stakeholder and objective. This process of alignment will continue for the next category: remaining.

Remaining	
To learn from failed projects	*Implement a post-mortem analysis* (Reel, 1999) If we didn't do this then we wouldn't learn from mistakes or successes. By studying how the project has been run many things can be learnt. (R5)

Table 6 categorisation of success factors in accordance with the category remaining

In section 2.4 it was established that the category remaining was not relevant to the framework as the factors encapsulated by this category concerned themselves with aspects outside the scope of the framework. It follows therefore that substantiating factors in this category can not endorse the framework. From table 6, however, it is apparent that the following success factor can be substantiated:

- *To learn from failed projects*

Aggestam (2001) informs that the framework does not propose to enlighten organisations with regard to this aspect it merely requires this. Indeed, it can be commented that the general tone interpreted from the material gathered in this report concerning this factor is supportive of Aggestam's (2001) observations. R5 offers a particularly concise motivation for a post-mortem analysis with the following quotation:

> If we didn't do this then we wouldn't learn from mistakes or successes. By studying how the project has been run many things can be learnt. (R5)

This success factor has been included for the simple fact that it is supportive of the point of view suggested by Aggestam (2001).It can therefore be commented that post-mortem analyses are equally valid in software projects that are not ISD related as they are in software projects that are ISD related. This can be further corroborated by information from section 2.3 where it was established that learning from failed projects was acknowledged as being relevant in both an IS and non IS context.

By comparing the factors identified in this report with those identified by Aggestam (2001) an association has been made with the framework and it can be commented that the two vital categories: stakeholder and objective have been substantiated. All factors pertaining to stakeholders can be confirmed whereas the factor: meet business objective, belonging to the category objective can not be substantiated. The explanation given for this discrepancy is that the domains of the two areas under discussion are different. It can therefore be commented that the framework is showing potential for use in software projects where the purpose is not to build an organisational IS.

Table 7 outlines all the success factors identified in this project and highlights the ones that align exactly with the success factors in the framework. This table will provide the foundation for the next stage of the analysis.

Framework	Success factor
Yes No No **Yes** **Yes** **Yes** No No	• *Well defined project objective* (Purba et al, 1995) • *Well defined project scope*(Purba et al, 1995; Field, 1997, in Reel, 1999) • *Minimise the scope of the project* (Johnson, 2001) • *Find a project sponsor /champion* (Johnson, 2001; The Standish Group, in Procaccino et al, 2002; Field, 1997, in Reel, 1999) • *Implement a post-mortem analysis* (Reel, 1999) • *Ensure effective communication between all parties* (Matheson and Tarjan ,1998; Purba et al, 1995) • *Incorporate effective software measurement into the process* (Paulish and Carleton, 1994; Capers Jones, 1995) • *Routines for effective quality control* –design reviews, code inspections-(Capers Jones, 1995)
No No	• *The use of a formal methodology* (Johnson, 2001 • *Use software tools* (Rakos, 1990; Capers Jones, 1995)
No No No No No	• *Include milestones* (Giglio, 1999; Capers Jones, 1995; Rakos, 1990) • *Set up realistic deadlines* (Field, 1997, in Reel, 1999; Glass, 1998b) • *monitor progress*(Capers Jones, 1995; Reel, 1999) • *Use estimating tools* (Capers Jones, 1995) • *Realistic estimations* (Johnson, 2001)
Yes No No No	• *Build the right team* (Reel, 1999) • *Keep attrition low* (Reel, 1999). • *Allocation of roles to team members* (Rakos, 1990; Constantine, in Rettig, 1990) • *Skills of the development team* (Krishnam, 1998; Duvall, 1995; Field,1997, in Reel, 1999)) • *Give the development team ownership* (Giglio, 1999; Rettig, 1990)
No **Yes** No	• *The ability to make good decisions* (Reel, 1999, Purba et al, 1995) • *Experienced project manager* (Johnson, 2001; Purba et al, 1995) • *Project managers have full authority to manage their project* (Procaccino et al, 2002)
Yes **Yes**	• *User involvement is necessary in the development process* (Amoako-Gyampah & white, 1997, in Procaccino et al, 2002; Giglio, 1999; Procaccino et al, 2002, Paulk, Curtis, Chrissis and Webster,1993, in Procaccino et al 2002) • *The user is knowledgeable and is authorised to make decisions* (Rakos, 1990).
No **Yes** No No No	• *Changes in requirements* (Duvall, 1995) • *Involve the user in the RE process* (Rakos, 1990; • *User resistance and communication* (Saiedian and Dale, 2000; Reel, 1999) • *Establish basic requirements* (Johnson, 2001) • *Complete and accurate requirements from the start*(Procaccino et al, 2002)

Table 7: table showing which factors align exactly with those factors in the framework

7.2 Analysis of the success factors not present in the current framework

By studying table 7 it becomes apparent that a multitude of success factors can not exactly be aligned with the success factors in the framework. This poses the natural question:

Can the framework be used in this type if project when so many factors cannot explicitly be aligned?

In order to answer this question, the first step would be to see if the remaining success factors can, in some other way, be grouped in accordance with the aforementioned categories; i.e. stakeholder and objective. The category remaining is not of interest here as this category does not endorse the framework. The remaining factors can not explicitly be aligned to the framework which would suggest that an implicit approach may be suitable.

As a starting point for this analysis it would seem prudent to reiterate some of the information previously outlined in section 2.4. This is best accomplished by once again referring to a diagram of the framework in figure 14

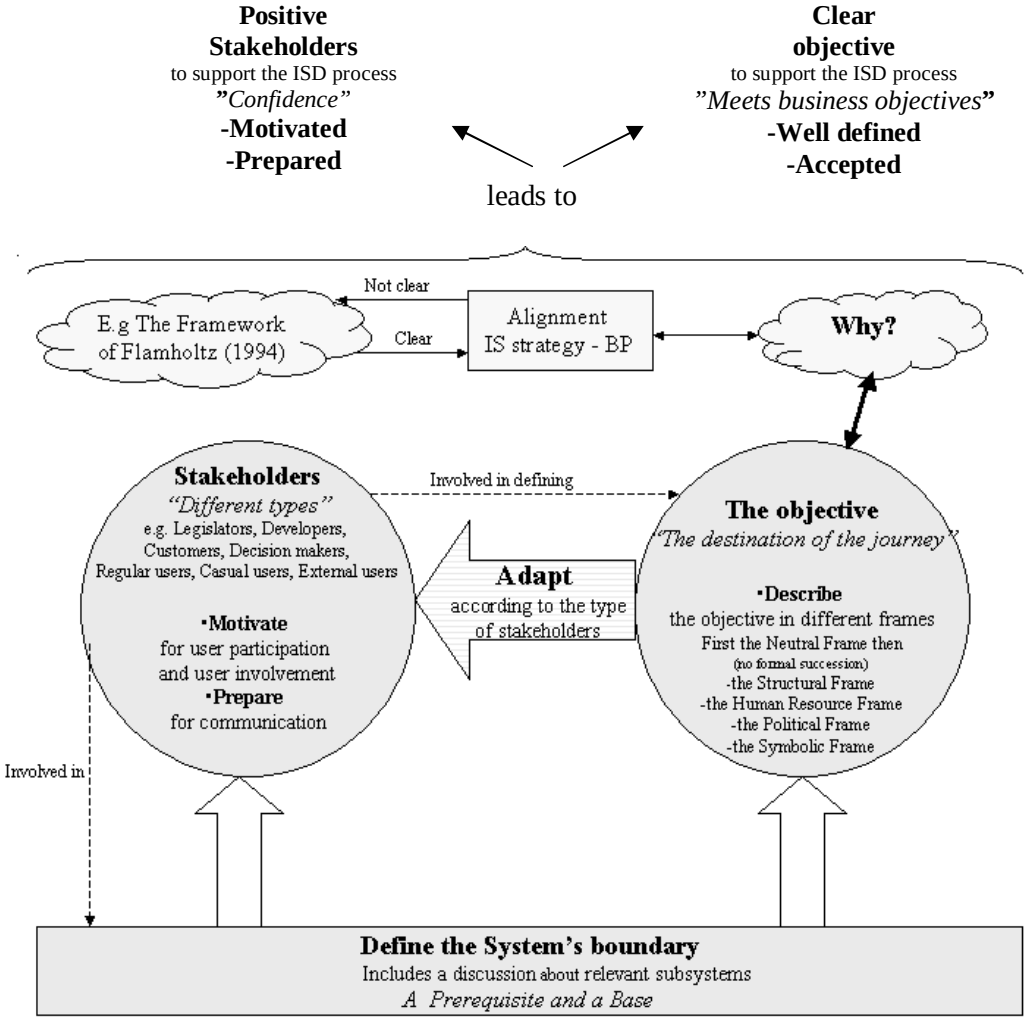

Figure 14: a framework for managing success factors in ISD projects (adapted from Aggestam, 2001, p. 64)

From figure 14 it once again becomes clear that in order to use the framework the system's boundary must be defined. The objective should then be described in different frames using different levels of abstraction. The objective should then be adapted to fit the stakeholders. The concept of adapting the objective was addressed by R1 who lends support to this concept with the following quotation:

> […] I believe the goal should be adapted for the person and the task that person is undertaking. (R1)

Aggestam (2001) maintains that using the framework leads to a clear objective and motivated stakeholders. Aggestam (2001) informs that a clear objective can be obtained by analysing the objective in different frames and different levels of abstraction (see section 2.4). In order to acquire motivated stakeholders Aggestam (2001) recommends that the motivation and preparation processes be followed.

According to Aggestam (2001) the *motivation process* should focus on the stakeholders' knowledge and needs. The motivation process aims to describe the objective in such a way that individual stakeholders understand why the project should be carried out and, more importantly, how the project will affect him/her. The *preparation process* aims to improve the communication between stakeholders by familiarising the stakeholders with the terms and concepts to be used within the project. Aggestam (2001) explains that the motivation and preparation processes aim to promote confidence and knowledge in the stakeholders about the prospect of change. Aggestam (2001) maintains that this is a prerequisite for stakeholder involvement and participation.

It has been established that the remaining success factors can not explicitly be aligned to the framework. Further, it has been suggested that an implicit approach may be suitable. A plausible strategy for implicitly aligning the remaining success factors would be to see if any of them encourage or support the preparation and motivation processes or if they assist in describing the objective. This will be attempted in the next sub-section.

7.2.1 Implicitly aligning remaining success factors to the categories stakeholder and objective

This sub-section will account for those success factors which have implicitly been aligned to the categories stakeholder and objective.

Stakeholder

- *The ability to make good decisions* (Reel, 1999, Purba et al, 1995
- *Give the development team ownership* (Giglio, 1999; Rettig, 1990)
- *User resistance and communication* (Saiedian and Dale, 2000; Reel, 1999)
- *The use of a formal methodology* (Johnson, 2001)
- *Realistic estimations* (Johnson, 2001)
- *Set up realistic deadlines* (Field, 1997, in Reel, 1999; Glass, 1998b)
- *Allocation of roles to team members* (Rakos, 1990; Constantine, in Rettig, 1990)

Aggestam (2001) informs that the ultimate goal of the motivation and preparation phases is to promote confidence and knowledge in the stakeholders about the prospect of change. It is the opinion of the author of this project that the ability to make good decisions could be a useful quality in achieving this purpose.

The motivation process according to Aggestam (2001) aims to describe the objective in such a way that individual stakeholders understand why the project should be carried out and, more importantly, how the project will affect him/her. Motivating why the success factor; give the team ownership is compatible in this context is aptly explained by R1:

> […] When the project starts it may be that not everyone agrees with the project plan […] we discuss it and arrive at a decision where everyone has had input. (R1)

Aggestam (2001) advises that the preparation process aims to improve the communication between stakeholders by familiarising the stakeholders with the terms and concepts to be used within the project. It is the opinion of the author of this project that the preparation process can only be achieved if the stakeholders are committed and do not resist change. This provides a tentative motivation for the validity of the success factor: user resistance and communication. R2 addresses the concept of user committance and ascertains that lack of commitment can jeopardise the projects success.

> […] usually it is harder to achieve a successful project without a committed user. (R2)

The following quotation from R1 establishes a connection to the motivation process thus implicitly aligning the success factor: use of a formal methodology to the category stakeholder:

> When you take on work you need to have a certain professionalism. You need, for example, to be able to promise that this and this will be delivered in 6 months and will cost X. You need to be able to guarantee certain things […] (R1)

It is the opinion of the author of this report that this quotation supports the motivation processes by considering the needs of the stakeholder, in this case the paying customer.

A further function of the motivation process is to clarify for the stakeholder how the project will affect him/her. As a customer paying for a product the accuracy and reality of the estimation is therefore very valid. The importance of the estimation is captured by R1 in the following quotation:

> If you have delivered within the allocated time and budget then it's a success. It's that simple. (R1)

Motivation for the alignment of the success factor: set up realistic deadlines to the category stakeholder can be found in the following excerpt provided by R3.

> It is these checkpoints where we agree that we can go further. We confirm the projects status with the sponsor and the customer. (R3)

By regularly informing the relevant stakeholder of the projects status must favour both the motivation and preparation processes in the sense that providing regular updates of the projects status must serve to promote confidence and knowledge in the stakeholders about the prospect of change.

The allocation of roles would certainly assist in promoting, in the stakeholders, confidence and knowledge for the prospect of change. This is effectively captured by R4 in the following quotation:

> […] if people don't know what their role or task is then they don't know what they should be doing and everything is uncertain. (R4)

The category stakeholder encompasses a variety of different individuals. From figure 14 it is apparent that examples of such individuals are given as: legislators, developers, customers and users to name but a few. It is the opinion of the author of this report that the success factors given below permit themselves for categorisation under stakeholder for the simple fact that they either relate directly to specific stakeholders or circumstances concerning stakeholders. It must however be stressed that this alignment is extremely weak as these success factors are neither compatible with existing factors in the framework nor the functionality of the framework. Therefore it must be concluded that these success factors do not in any way support the framework.

- *Skills of the development team* (Krishnam, 1998; Duvall, 1995; Field,1997, in Reel, 1999)
- *Keep attrition low* (Reel, 1999).
- *Project managers have full authority to manage their project* (Procaccino et al, 2002)

Seven additional success factors have implicitly been aligned to the category stakeholder. Although this alignment is not as strong as the explicitly aligned factors outlined in sub-section 7.1.1 there is nonetheless a correlation between these implicit success factors and the inner workings of the framework. Three additional factors have lent themselves to categorisation under the stakeholder category for the simple fact that they are directly relatable in that context. These factors as a consequence of this weak alignment do not in any way substantiate the framework.

Objective

Aggestam (2001) informs that a clear objective can be obtained by analysing the objective in different frames and different levels of abstraction. In other words, the frames provide a certain perspective when viewing the objective. These frames and levels of abstraction have been reviewed in section 2.4 but will be reiterated here in order to provide clarity in the analysis:

- *The structural frame:* in this frame the structure of the organisation is viewed
- *The human resource frame:* in this frame the relationship between people and the organisation is viewed
- *The political frame:* this frame views the importance of handling conflicts, disputes, power and decisions in organisations
- *The symbolic frame:* this frame views symbols as embodying and expressing an organisations culture.
- *The neutral frame:* this frame views the neutral perspective of the organisation. It focuses on aspects such as business mission, plan and size.

In each frame the following three levels of abstraction are considered:

- **What:** what do we need

- **Why:** why do we need it
- **How:** how are we going to implement it

It has been well established that the framework has been designed with the purpose of assisting ISD projects in their planning stages. It has further been established by McDermid (1990) that the domain of the problem to be addressed constitutes a fundamental difference between the domains of SE and ISD. Avison and Shah (1997) remind that the IS should support the organisation it is implemented in. The frames outlined above consequently provide a very detailed mechanism for enabling this. It is, however, unlikely that an embedded system or a specific PC application is required to support the organisation it is implemented in to the same degree as an IS and therefore it is not certain that defining the objective in all the frames is necessary.

It is the opinion of the author of this report that implicitly aligning the remaining success factors to the category objective using the frames or perspectives as a point of reference can not really be attempted. This can be explained simply by the fact that the frames, or perspectives, provided by Aggestam (2001) have been included in the framework as they encourage or promote valid viewpoints when planning an IS. The validity of these frames in an SE perspective is surrounded by question marks and uncertainties. In other words, using these frames as a point of alignment is purely speculative. However, if the frames were to be disregarded in this context and concentration was focused on the levels of abstraction then the author of this report feels that an implicit alignment can be undertaken.

As is highlighted by Aggestam (2001) the objective, in each frame, should be analysed using the levels of abstraction: what, why and how. By disregarding the frame and instead using the levels of abstraction as a foundation the following success factors can be implicitly aligned to the "what" level of abstraction:

- *Establish basic requirements* (Johnson, 2001)
- *Complete and accurate requirements from the start* (Procaccino et al, 2002)

Requirements constitute one of the fundamentals of a project. Without them a product can not be realised. R2 clarifies this in the following quotation:

> […] one needs to evaluate exactly what we need in order to build a function […] (R2)

Whether one chooses to identify complete requirements or basic requirements is irrelevant. The issue under scrutiny here is that which is required.

Using a similar strategy the following success factors can implicitly be aligned to the "how" abstraction

- *Changes in requirements* (Duvall, 1995)
- *Incorporate effective software measurement into the process* (Paulish and Carleton, 1994; Capers Jones, 1995)
- *Routines for effective quality control* –design reviews, code inspections- (Capers Jones, 1995)
- *Use software tools* (Rakos, 1990; Capers Jones, 1995)
- *monitor progress*(Capers Jones, 1995; Reel, 1999)
- *Minimise the scope of the project* (Johnson, 2001)

The “how” level of abstraction serves to ascertain the means by which something is to be implemented. The following quotation provided by R3 motivates how managing changes to requirements contributes to this:

> You verify the requirements with the customer. If you change something you get another product. When you verify later with the customer and discover that changes have been made it can be very expensive compared to what it would have been if you had managed the changes correctly. (R3)

In a similar way the success factor: incorporate effective software measurement into the process can be aligned to the “how” level of abstraction. This success factor incorporates features such as lines of code, scheduled time and man hours. The following excerpt provided by R2 endorses the relevance of this success factor in the context given even if only the time aspect is specifically addressed:

> […] Often time is something which is underestimated. I have, on several occasions, witnessed that competence is required when estimating time. Especially when giving a price tag. (R2)

R3 and R5 endorse the validity of quality control and how they influence implementation in the following excerpts.

> These things belong to the process. They have to be there […](R3)

> We must have routines before we deliver […] (R5)

The following quotation from R2 provides a concise motivation for implicitly aligning the use of software tools to the “how” abstraction:

> It (the project) would be more expensive and the quality would suffer. (R2)

Monitoring progress aptly lends itself for alignment to the “how” abstraction with help of the following quotation:

> It’s these checkpoints where we agree that we can go further […] (R3)

The success factor: minimise the scope of the project cannot be corroborated with any excerpt from the interview material. However, it is the opinion of the author of this report that this factor can be aligned to the “how “abstraction as minimising the scope of the project is very suggestive of how the objective can be realised.

An in depth analysis of the gathered material has now been given. The chapter will continue, and conclude, with the conclusions and indeed result of this report.

7.3 Conclusions

From the previous section it has been established that the critical categories of the framework: stakeholder and objective have, through explicit alignment to existing success factors, been substantiated. Figure 15 graphically portrays this.

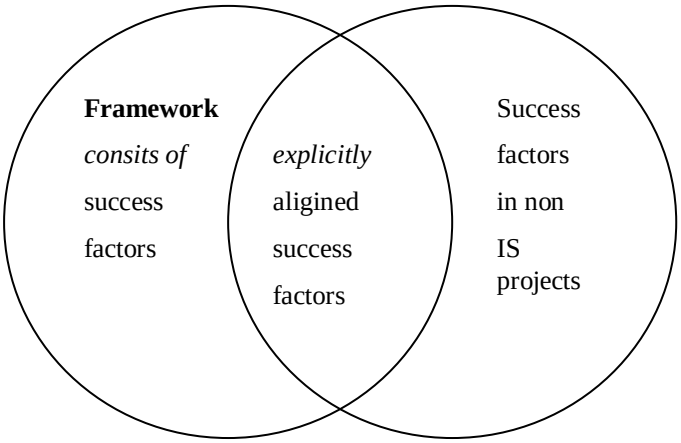

Figure 15: diagram showing compatibility of success factors

It has further been shown that the majority of the success factors identified in this report have not been aligned in this explicit manner. It has also been shown that three factors lie outside of the framework. A detailed presentation and account for the achievements of this project will now follow.

Table 8 lists those explicit success factors and the categories to which they have been aligned.

category	Explicit success factor
Objective	*Well defined project objective* (Purba et al, 1995)
Stakeholder	*Find a project sponsor /champion* (Johnson, 2001; The Standish Group, in Procaccino et al, 2002; Field, 1997, in Reel, 1999)
Remaining	*Implement a post-mortem analysis* (Reel, 1999)
Stakeholder	*Ensure effective communication between all parties* (Matheson and Tarjan ,1998; Purba et al, 1995)
Stakeholder	*Build the right team* (Reel, 1999)
Stakeholder	*Experienced project manager* (Johnson, 2001; Purba et al, 1995)
Stakeholder	*User involvement is necessary in the development process* (Amoako-Gyampah & white, 1997, in Procaccino et al, 2002; Giglio, 1999; Procaccino et al, 2002, Paulk, Curtis, Chrissis and Webster,1993, in Procaccino et al 2002
Stakeholder	*The user is knowledgeable and is authorised to make decisions* (Rakos, 1990).
Stakeholder	*Involve the user in the RE process* (Rakos, 1990;

Table 8: explicit success factors

It has further been established that there exists a number of success factors that can not explicitly be aligned to the factors in the framework. An attempt to implicitly

align these factors to the categories stakeholder and objective has been conducted and it has been shown that further success factors can be aligned to the framework using this approach. Table 9 offers an overview of these implicitly aligned success factors and the categories to which they have been aligned. It can be commented that all factors that have been implicitly aligned to the objective have been aligned to a specific level of abstraction. This has been indicated by writing (how) or (what) beside the word objective.

category	Implicit success factor
Stakeholder	• *The ability to make good decisions* (Reel, 1999, Purba et al, 1995
Stakeholder	• *Give the development team ownership*(Giglio, 1999; Rettig, 1990)
Stakeholder	• *User resistance and communication* (Saiedian and Dale, 2000; Reel, 1999)
Stakeholder	• *The use of a formal methodology* (Johnson, 2001)
Stakeholder	• *Realistic estimations* (Johnson, 2001)
Stakeholder	• *Set up realistic deadlines* (Field, 1997, in Reel, 1999; Glass, 1998b)
Stakeholder	• *Allocation of roles to team members* (Rakos, 1990; Constantine, in Rettig, 1990)
Objective (what)	• *Establish basic requirements* (Johnson, 2001)
Objective (what)	• *Complete and accurate requirements from the start* (Procaccino et al, 2002)
Objective (how)	• *Changes in requirements* (Duvall, 1995)
Objective (how)	• *Incorporate effective software measurement into the process* (Paulish and Carleton, 1994; Capers Jones, 1995)
Objective (how)	• *Routines for effective quality control* –design reviews, code inspections-(Capers Jones, 1995)
Objective (how)	• *Use software tools* (Rakos, 1990; Capers Jones, 1995)
Objective (how)	• *monitor progress*(Capers Jones, 1995; Reel, 1999)
Objective (how)	• *Minimise the scope of the project* (Johnson, 2001)

Table 9: implicit success factors

There remain, however, a few success factors that can neither explicitly nor implicitly be aligned to the framework. These factors are listed in table 10.

framework	Success factors that cannot be aligned to the framework
No	*Skills of the development team* (Krishnam, 1998; Duvall, 1995; Field,1997, in Reel, 1999)
No	*Keep attrition low* (Reel, 1999).
No	*Project managers have full authority to manage their project* (Procaccino et al, 2002)

Table 10: success factors that neither explicitly nor implicitly support the framework

These three success factors have shown themselves to lie outside the scope of alignment are thus incompatible for the framework. A plausible suggestion for this non alignment could be the fact that these particular factors are related to the *development* process. The focal point of the framework is the *planning* stage of the IS and thus would not, and does, not consider factors that are compatible to other aspects of the process.

Figure 16 graphically portrays the accomplishments of this report. An explicit alignment to the heart of the framework has been achieved. The framework has been founded on central success factors, the heart metaphor is used to signify this. These success factors have been categorised in accordance with the groups: stakeholder and objective. By explicitly aligning the material identified in this project to these success factors a confirmation for the framework's potential to be applied in software projects where the purpose is not to build IS can be deduced.

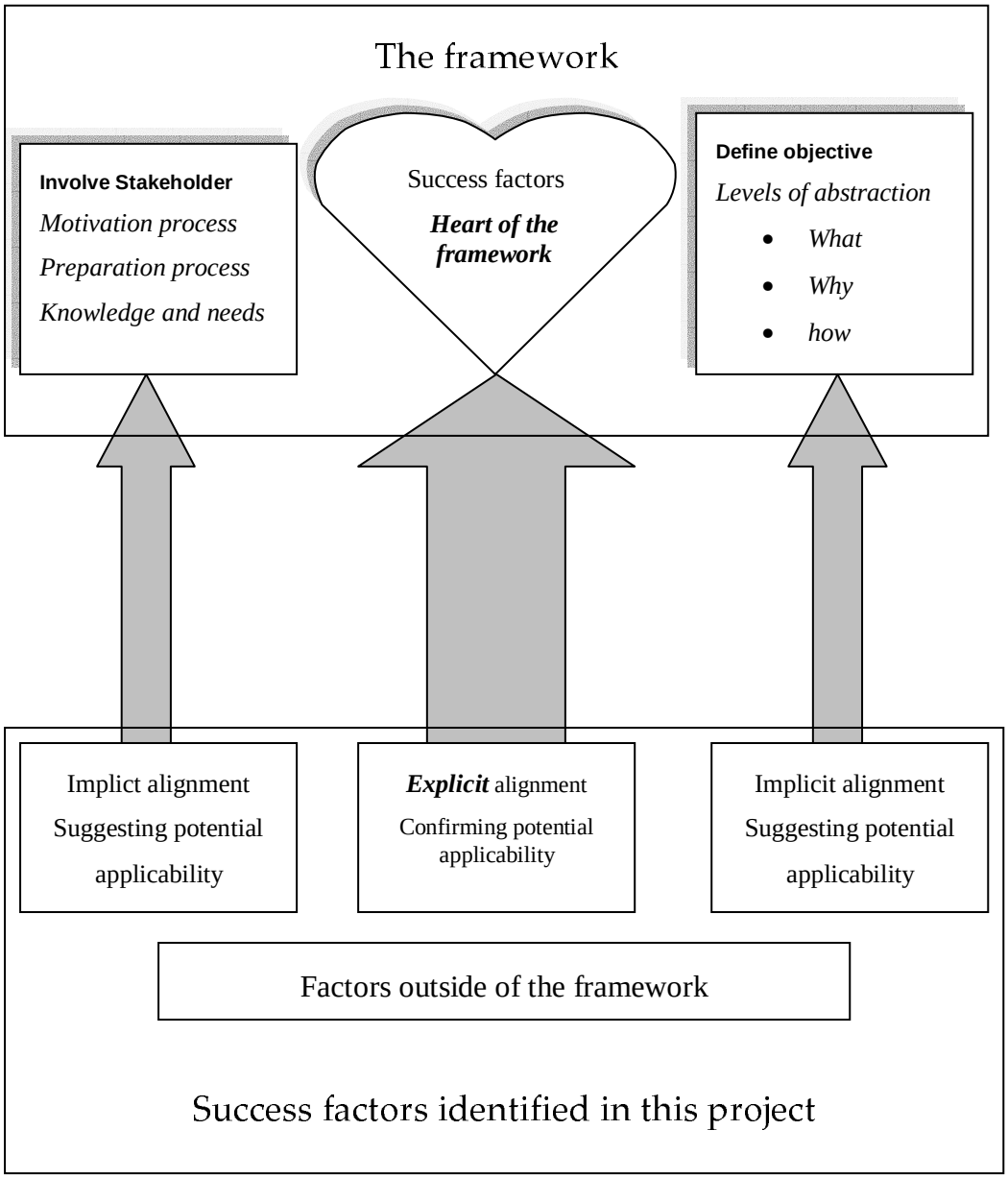

Figure 16: the accomplishments of this report

Explicitly aligning material in this report to the framework offers a powerful motivation for the framework's applicability in non IS software projects. However, a large number of success factors identified in this report have not explicitly been aligned to the framework (see table 7, p. 86) thus challenging the strength of the framework's applicability in this type of project. By implicitly aligning a number of success factors (see table 9) to the inner workings of the framework i.e. the motivation

and preparation processes and the what, why and how levels of abstraction has given further indication of the framework's potential in this context. The fact that this implicit alignment merely suggests potential is not as strong as the explicit alignment and is therefore something which needs to be considered when answering the question originally formulated in section 3.1:

Does the framework show potential for application in software projects where the purpose is not to build an organisational information system?

By explicitly aligning all, with the exception of one, success factors in the framework to success factors in this report confirmation of the framework's potential for application in non IS software projects has been achieved. The implicitly aligned success factors merely suggest potential for use but nothing more. Additionally, the number of implicitly aligned factors outweighs the explicitly aligned factors thus raising the question:

Can the framework be used in this type if project when so many factors cannot explicitly be aligned?

Before this question is answered it can be commented that the success factors that have implicitly been aligned to the category stakeholder through the motivation and preparation processes are more firmly anchored than those that have been aligned to the three levels of abstraction. This is explained by the fact that describing the objective requires that the frames be used *in addition* to the levels of abstraction. As has already been highlighted, the frames have not been used to implicitly align the material in this report to the category objective.

That which with all certainty can be ascertained is that the work in this report has certainly contributed with the first stages of expanding the framework into newer areas of application. The work carried out has *not*, however, been sufficient in order to state that the framework *can* be used in software projects where the purpose is not to build an organisational IS.

A presentation of the result of this report has now been given and thereby an answer to aim has been offered. The completion of the analysis and presentation of the result constitutes completion of the fourth objective outlined in section 3.1:

- *Evaluate the material gathered in the field work*

In chapter 8 the material gathering process and the result of the analysis will be critically reviewed.

8 Concluding remarks and future work

Berndtsson et al (2002) inform that once the result has been presented the next step is to consider, or discuss how these results fit into the greater picture of the chosen area of study. In order to do this Berndtsson et al (2002) recommend that one should consider a number of aspects, amongst other things: any contributions that can be made to research, how the results compare with those of others etc. In addition to this context discussion Berndtsson et al (2002) inform that a critical review of the work carried out is also relevant in order to bring to the surface the strengths and weaknesses of the project. A further point to consider is the identification of future work which thus offers a point of continuation for any further research.

The structure of this section is as follows: in section 8.1 a critical review of the material gathering process will be given. This section will concentrate solely on the issues raised and described in chapters 5 and 6. Section 8.2 will attempt to place the findings of this report in the context of the area of study and will consider, amongst other things, what contributions have been made to the chosen field of research. The chapter will conclude with a presentation of suggestions for future work.

8.1 Critical review of the material gathering process

The purpose of this section is to provide a critical review of the material gathering process. This will be accomplished in two sub-sections where the first will concern itself with the data gathering process, i.e. the literature analysis and the interview survey. The second sub-section will concern itself with the analytical process, i.e. how the data was interpreted and how the result was formed.

8.1.1 The data gathering process

The literature analysis will now be reviewed.

The literature study

The following quotation motivates the use of a literature analysis in the context of this report:

> *"The use of a literature analysis in the context of this report has been motivated by the fact that the information obtained from such a study would provide a stable base from which a further, deeper study could then be conducted."* (p. 41)

It is the opinion of the author of this report that the literature analysis has fulfilled this purpose excellently thereby pinpointing the study's main strength. Additional strengths are considered as being the variation and breadth of the material. Further it can be commented that the above quotation constitutes a valid motivation for the completeness of the literature study. Berndtsson et al (2002) address the issues of validity and reliability in literature analyses by referring to topics such as completeness and identification of relevant sources. The concepts of validity and reliability are very relevant when critically reviewing ones chosen approach to work.

Berndtsson et al (2002) inform that knowing when enough material has been gathered is difficult to ascertain. However, if a clear strategy has been conveyed Berndtsson et al (2002) advise that the completeness of a literature analysis can easily be identified and understood by a potential reader. By describing the purpose of the literature

analysis and then showing that it has fulfilled this purpose is considered by the author of this report to be clear evidence for the completeness of this particular literature analysis.

Berndtsson et al (2002) remind that the identification of relevant sources constitutes an important factor in the context of validity and reliability. The literature analysis conducted in this report has rested heavily on the seven categories of software project success identified in Procaccino et al (2002). These categories constitute an additional strength in the sense that they provide an effective delimitation. The seven categories identified by Procaccino et al (2002) are given as: management, customers and users, requirements, estimation and scheduling, the project manager, the software development process, development personnel. As can be seen, these categories cover a large area. Using this categorisation has enabled identification of material spanning over this large area thus satisfying the condition: identification of relevant sources.

A potential weakness of the literature analysis is the quality of some of the gathered material. As is highlighted by Dawson:

> *"Literature is presented in a number of different formats. Some forms are more accessible than others and some are recognised as being more academically valuable and worthy. "*(Dawson, 2000, p. 67)

Dawson (2000) explains that material identified in the literature search should be recognised. This simply means that material has been assessed for its academic worth by other experts. It is not the case that all the literature in this study has undergone this assessment. Erikson (1999) advises that one should be restrictive when referring to magazines and other periodicals as these sources are usually not peer reviewed. Although the literature analysis is not built entirely on such articles the reference list does contain examples of this type of article.

The interviews

In all, five interviews were conducted at local software companies and all respondents were met in their working environments. Patel and Davidson (1994) address the issues surrounding reliability and validity in the context of interviews and the issues surrounding these phenomena. The reliability of the interview study, according to Patel and Davidson (1994) is influenced by the *interview effect* which can arise. Andersen and Schwencke (1998) inform that the interview effect is that effect or influence the interviewer has over the respondent.

Judging the severity of the effect one has had over the respondents is not an easy task. It is the opinion of the author of this report that two factors, initiated in this report, can be highlighted which may have assisted in minimising this effect.

- All respondents were willing to answer questions
- The respondents were met face to face thus enabling the interviewer to record the respondents reactions to the questions

Patel and Davidson (1994) advise that it is important to show a genuine interest for the person one is interviewing. The author of this report feels that a good rapport was established with all respondents hence their willingness to answer questions. Andersen and Schwencke (1998) inform that the interviewer should record the respondents' reactions during the session as this can provide useful information. In the case of the interviews carried out in this study no emphasis was placed on

recording these reactions although the interviewer was able to, and did observe the respondents reactions when each question was asked. The interviewer did not, in any of the interview sessions, interpret any kind of discomfort; uncertainty or fear from any of the respondents.

Berndtsson et al (2002) highlight that open interviews, when mastered properly, have the potential to address issues of real importance and thus provide very valuable information. This type of interview technique was adopted for this report and is considered by the author of this report to have been successful in gathering a rich and varied material. A potential weakness of these sessions is encapsulated by the issue of validity. Berndtsson et al (2002) explain that *researcher bias* is an aspect that requires careful consideration in the context of open interviews. Berndtsson et al (2002) explain that research bias is explained by the researcher's pre-conceptions and a priori theories which may affect the outcome of the study. It is the opinion of the author of this report that the authors own willingness to find a correlation between the framework and the gathered material has, in some ways, showed itself in the formulation of the interview questions. This can be explained by the fact that some questions have begun with the formulation: *"how important is the aspect…"* or ended with the formulation: *"…do you consider this to be important"*. These formulations automatically reflect the opinions of the interviewer and thus may provoke an answer which favours that impression.

A further weakness may be found in the limited scope of the interview survey. Two software companies have been represented by the survey and therefore the issue of *generalisation* requires consideration. Patel and Davidson define generalisation as :

> *Is the result valid for other individuals than those covered by the survey.* (Patel and Davidson, 1994, p. 43)

It is the opinion of the author of this report that the limited scope of the interview survey can not possibly be representative of the views of all software companies. This could possibly be rectified by interviewing more respondents from a wider range of software companies. However, this approach assumes that one has certain resources at their disposal. In chapter 4 five elements were identified by Dawson (2000) as being relevant when conducting a project, these elements are reiterated here:

- Time
- Cost
- Quality
- Scope
- Resources

It is the opinion of the author of this report that due to severe time limitations nothing more could have been done to expand the scope of the interview survey. Furthermore, due to the fact that few software companies were identified in the local area a large amount of travelling would have been required to involve other companies. This challenges the cost factor, which is of particular relevance as this report has been carried out on severely limited funds. Naturally, telephone interviews could have been conducted as a way of reducing costs but this subsequently raises the issue of interview quality, as the respondents' reactions cannot be observed. It can, however, be commented that telephone interviews were considered but no respondents were forthcoming.

A critical analysis of the data gathering process has now been given. The discussion will continue with a critical review of the analytical process.

8.1.2 The analytical process

The material gathered throughout the course of this report has been presented in chapter 6 and analysed in chapter 7. The raw material and subsequent analysis have formed the foundation for the result of this work and thereby an answer to the aim of the report has been offered. The issues of validity and reliability have already been addressed in the context of the data gathering process. It is the opinion of the author of this report that these issues are equally relevant in the analytical process.

Patel and Davidson (1994) address the issue of *simultaneous validity* and explain that this is a way of establishing if the same result obtained from one technique can be obtained by applying another technique. It has been established that the literature analysis has provided, through the identification of success factors, a stable base upon which a deeper interview study has been built. When conducting the analysis the interview material was weighed up against the success factors gathered in the literature analysis. Material was then extracted from the interviews which corroborated the success factors; i.e., confirmed that these factors were indeed a success. This is considered by the author of this report to be a valid motivation for ensuring validity of the analytical process.

The issue of research bias is an important issue to be addressed in the analytical process. Andersen and Schwencke (1998) highlight that it can be very tempting to integrate ones own opinions and points of view into the analytical process so that the material better agrees with ones own point of view. However, as is recommended by Andersen and Schwencke (1998) one should, in all research work, maintain an honest and honourable relationship to the information one has gathered. The author of this report admits to, at times, having suffered the temptation to add own inflections to the material but has nonetheless withheld a respectable distance to the material and not adjusted it in any way .

A critical review has now been provided for both the data gathering and analytical processes. The discussion will now shift focus and attempt to put the result of the report into a wider perspective.

8.2 Putting the results into context

Berndtsson et al (2002) maintain that it is important to discuss how the results of a report fit into the wider picture of the chosen area of study. Berndtsson et al (2002) provide assistance in this difficult process by offering a number of questions, of which the answers to may provide help when formulating this section. It is the opinion of the author of this report that these questions provide an excellent mechanism for structuring this section.

What can my results be used for?

McConnell, 1996, in Procaccino, Verner, Overmyer & Darter (2002) informs that 20% of software projects are failures and 46% experience cost and schedule overruns. Hoffman, 1999, in Procaccino et al, 2002) ascertains that failure rates for software development projects are as high as 85%.

These comments provide the initial starting point for justifying the relevance of this work and what it may possibly contribute with. It has been established in section 2.4 that a framework exists for ISD projects where the purpose is to build an organisational IS. These IS projects are also plagued by astonishing failure rates. The framework proposed by Aggestam (2001) aims to rectify the failure rate of ISs by guiding organisations in what considerations need to be made before the IS project begins.

The aim of this report proposed to investigate the applicability of the framework in software projects where the purpose was not to build an organisationl IS. By satisfying this aim it is hoped that this report can contribute with material that may prove useful when considering how the success rate of non IS software projects can be improved.

Have I made a contribution to my field of research?

The work carried out in this report has attempted to extend the framework's scope of applicability by considering software projects where the purpose is not to build an organisational IS. The framework's potential for use in this type of project has been confirmed by explicitly aligning material identified in this report to success factors in the framework. It has further been ascertained that only a few of the identified success factors have been aligned in this manner (see table 7), thus challenging the framework's potential for use in this type of project. By implicitly aligning a majority of the remaining success factors to the inner workings of the relationship; i.e. the motivation and preparation processes further strength for the framework's use in this context has been acquired.

The work in this report has certainly contributed with the first stages of expanding the framework into newer areas of application. The work carried out has not, however, been sufficient in order to state that the framework *can* be used in software projects where the purpose is not to build an organisational IS.

8.3 Future work

In section 8.2 useful tips were provided by Berndtsson et al (2002) concerning how best to set the results in the context of the bigger picture. In a similar way, Berndtsson et al (2002) offer valuable tips for identifying future work. These tips, also in question form, will be used as a way of structuring this section.

If I had more time to devote to my report what would I do?

If the report had more time at its disposal the questionnaire survey would have been reattempted on a much larger scale. Naturally, one could conduct more interviews but these are both time consuming and tedious. Although there is always a risk for a low response rate in questionnaire based surveys one reaches out to a much wider population thus enabling the possibility to draw more general conclusions.

The result, in its current state, merely suggests a strong potential for using the framework in non IS software projects. In order to develop this result into a more specific statement it is necessary to establish guidelines for *how* the framework could be used in this type of project.

What still needs to be done before my results can be applied in practice?

The factors that have implicitly been aligned to the framework need to be more thoroughly worked with in order to generate a more explicit alignment to the framework. At present these factors have been aligned to certain processes in the framework; i.e. the motivation process, the preparation process and different levels of abstraction. In order to establish a firmer alignment it is necessary to investigate the relevance of these processes from the perspective of non IS software projects. The following questions may provide help when considering this:

- Do the implicitly aligned success factors support the motivation and preparation processes?
- Are these processes at all relevant when conducting a software project where the purpose is not to build an organisational IS?

Answering these questions should improve the reliability of the implicit alignment conducted in section 7.2.1 thus generating a more explicit alignment. It has been commented that the factors implicitly aligned to the category objective are not as firmly anchored as those which have been aligned to the category stakeholder. This is explained by the fact that the success factors have merely been aligned to levels of abstraction. Aggestam (2001) states specifically that the objective should be described in different frames, or perspectives, *as well* as the different levels of abstraction. As has been established, the frames have been completely disregarded when implicitly aligning material to the category objective. This poses the following question:

- What relevance do the frames: structural, human resources, political, symbolic and neutral have in the context of non IS software projects?

Once the implicit success factors have been more explicitly aligned a case study can be conducted in order to validate the framework's applicability in a software project where the purpose is not to build an organisational IS.

Suggestions for continued work have been offered and it is the opinion of the author of this project that a stable groundwork has been laid for any future research that may be done to apply the framework in non IS software projects. The completion of this final section concludes this project and thus the completion of the fifth objective:

- *Complete the report*

References

Aggestam, L. (2001) *Planning for Information Systems Development- A framework for supporting the management of Success Factors.* Dissertation for the degree of M.Sc: University of Skövde.

Andersen, E, S. (1994) *Systemutveckling- principer, metoder och tekniker*. Lund: Studentlitteratur.

Andersen, E. S. & Schwencke, E. (1998) *Projektarbete- en vägledning för studenter*. Lund: Studentlitteratur.

Andriole, S. J. & Freeman, P. A. (1993) Software systems engineering: the case for a new discipline. I: M. Dorfman & R. H. Thayer (red: er), Software Engineering (s. 29-43). Los Alamitos: IEEE Computer Society Press.

Avison, D. E. and Fitzgerald, G. (1993) *Information Systems Development- Methodologies, Techniques and Tools*. Henley on Thames: Blackwell Scientific Publications.

Avison, D. & Shah, H. (1997) *The information systems development life cycle: A first course in information systems.* Berkshire: McGraw-Hill Book Company.

Berndtsson, M. Hansson, J. Olsson, B. & Lundell, B. (2002) *Planning and implementing your final year project with success!* Gateshead: Springer-Verlag London Limited.

Boehm, B. W. (1988) A Spiral Model of Software Development and Enhancement. I: M. Dorfman & R.H. Thayer (red: er), Software Engineering (s. 415-426). Los Alamitos: IEEE Computer Society Press.

Brown, A. W. Earl, A. N. & McDermid, J. A. (1992) *Software Engineering Environments Automated Support for Software Engineering*, Berkshire: McGraw-Hill Book Company Europe.

Capers Jones Software Productivity Research (1995, March) Patterns of Large software systems: Failure and success, *Computer.*

Comer, E. R. (1997) Alternative Software Life Cycle Models. I: M. Dorfman & R. H. Thayer (red: er), *Software Engineering* (s. 404-414). Los Alamitos: IEEE Computer Society Press.

Dawson, C. W. (2000) *The essence of Computing projects A Student's Guide*. Harlow: Pearson education Limited

Dorfman, M. & Thayer, R. H. (1997) preface I: M. Dorfman & R. H. Thayer (red: er), Software Engineering (s. ix). LosAlamitos: IEEE computer Society Press.

Duvall, L. M. (1995) A Study of Software Management: The State of Practice in the United States and Japan. *The journal of Systems Software, 31,* pp. 109-124.

Erikson, M. G. (1999) *Att skriva litteraturreferenser enligt Harvard-systemet*. Institution för datavetenskap: Högskolan i Skövde

Fox, M. J. (1982) *Software and its development.* Englewood Cliffs, N.J, USA: Prentice Hall, Inc.

Frey, J. H. & Oishi, S. M. (1995) *How to conduct interviews by telephone and in person*. USA: Sage Publications, Inc.

Giglio, C. M. (1999, 25 January) Looking for a better way to develop software? Follow these ten commandments, *Infoworld.*

Glass, R. L. (1999) Evolving a New Theory of Project Success. *Communications of the ACM, 42*, (11).

King, D. (1988) *Creating effective software*. New Jersey: Prentice Hall.

Krishnan, M. S. (1998) The role of team factors in software cost and quality An empirical analysis, *Information Technology and People, 11*, pp. 20-35.

Leibowitz, J. (1999) Information Systems: Success or Failure?, *Journal of Computer Information Systems*

Johnson, J. H. (2001, February) Micro Projects Cause Constant Change, Exert from Extreme CHAOS published by The Standish Group

Jones, G. W. (1990) *Software Engineering*. Canada: John Wiley & sons Inc.

Judd, C. M., Smith, E. R. & Kidder, L. H. (1991) *Research Methods in Social Relations.* Fort Worth: Harcourt Brace (sixth edition).

Matheson, L. R. & Tarjan, R. E (1998, fall) Culturally Induced Information Impactedness: A Prescription for Failure in Software Ventures, *Journal of Management Information Systems.*

McDermid, D. C. (1990) *Software Engineering for Information Systems*. Worcester: Blackwell Scientific Publications.

Norris, M. & Rigby, P. (1992) *Software Engineering Explained*. Chichester: John Wily & Sons Ltd.

Patel, R. & Davidson, B. (1994) *Forskningsmetodikens grunder Att planera, genomföra och rapportera en undersökning* Lund: Studentlitteratur (Andra upplagen*).*

Paulish, D. J. & Carleton, A. D. (1994, September) Case Studies of Software-Process-Improvement Measurement, *Computer*.

Paulk, M. C. Curtis, B. Chrisis, M. B. & Weber, C. V. (1993) The Capability Maturity Model for Software. I: M. Dorfman & R.H. Thayer (red:er), Software Engineering (s. 427-438). Los Alamitos: IEEE Computer Society Press.

Pressman, R. S. (1997a) Software Engineering. I: M. Dorfman & R. H. Thayer (red:er), *Software Engineering* (s.57-74). Los Alamitos: IEEE Computer Society press.

Pressman. R. S. (1997b) *Software Engineering-a guide for practioners*. USA: McGraw-Hill Companies, Inc.

Procaccino, J. D. Verner, J. M. Overmyer, S. P. & Darter, M. E. (2002) Case study: factors for early prediction of software development success. *Information and Software Technology, 44*, pp. 53-62.

Purba, S. Sawh, D. Shah, B. (1995) *How to Manage a Successful Software Project Methodologies Techniques Tools*, USA: John Wiley & Sons, Inc.

Rakos, J. (1990) *Software project Management for small to medium sized projects*. Englewood Cliffs, New Jersey: Prentice Hall, Inc.

Reel, J. S. (1999,May/June) Critical Success Factors In Software Projects, *Computer*.

Repstad, P. (1994) *Närhet och distans Kvalitativa metoder I samhällsvetenskap*. Lund: studentlitteratur.

Rettig, M. (1990) Software Teams. *Communications of the ACM, 33*(10).

Révay, P. (1992) *Modern Systemförvaltning*. Lund: Studentlitteratur.

Sage, A. P. & Palmer, J. D. (1990) Software Systems Engineering. Canada: John Wiley & Sons, Inc.

Saiedian, H. & Dale, R. (2000) Requirements engineering: making the connection between the software developer and customer, *Information and Software Technology, 42*, pp. 419-428

Wateridge, J. (1997) how can IS/IT projects be measured for success?, *International Journal of Project Management, 16*, pp. 59-63.

Wood Harper, A. T. Antill, L. Avison, D. E. (1985) *Information Systems Definition: The Mulitview Approach.* Oxford: Blackwell Scientific Publications.

Yeo, K. T. (2002) Critical failure factors in information system projects, *International Journal of project Management, 20*, pp. 241-246.

Zikya Aktas, A. (1987) *Structured analysis & design of information systems*. New Jersey: Prentice-Hall, Inc.

Vill ni vara snälla och hjälpa mig?

Jag behöver Er hjälp med att få del av Era åsikter om det som kännetecknar ett ”lyckat” mjukvaruprojekt.

Jag heter Timothy Little och går sista terminen på det systemvetenskapliga programmet vid Högskolan i Skövde. Under denna termin arbetar vi individuellt med våra examensarbeten vilket innebär att vi fördjupa oss i ett område som vi själva har valt. Ett mycket viktigt inslag i mitt arbete är att undersöka hur professionella mjukvaruutvecklare tolkar och värderar de faktor som tros ligga till grund för ett lyckat mjukvaruprojekt.

Mycket kan hämtas från litteraturen men verkliga åsikter är mycket svårare att anskaffa. Det är detta jag behöver Er hjälp med. Om Ni tar Er tid till att besvara den bifogade enkäten så kommer Ni att hjälpa mig mycket i mitt arbete.

Enkäten behandlas givetvis helt konfidentiellt.

Vill Ni vara snälla och svara på enkäten elektroniskt eller på vanligt sätt **senast den 10 april**.

Tack på förhand

Timothy Little

Ev. frågor besvaras av Timothy Little:

Telefon: 0500-43 65 09

Mail: a00timli@student.his.se

Adress: Timothy Little

Södra Trängallén 7C

541 46 SKÖVDE

Supplement 2: questionnaire

ENKÄTUNDERSÖKNING

Om raderna ej räcker till, så fortsätt med Era svar sist i enkäten eller, om Ni skriver ut enkäten, använd baksidan. Glöm ej att ange frågans nummer.

1. ALLMÄNNA FRÅGOR

Då alternativ finns, markera med kryss det alternativ som stämmer i just ert fall

1.Vilken typ av mjukvara tillverkar Ni? _____

_____

_____

2. Antal anställda: _____

Mjukvaruutvecklingsprocessen

3a. Använder Ni en specifik metodologi under utvecklingsprocessen?

_____ JA

_____ NEJ

Om ja,vilken metodologi(t.ex.RAD,SSADM)?_____

_____

_____

3b Anser Du det vara nödvändigt att använda en metodologi för att få ett *"lyckat"* mjukvaruprojekt?

____ JA

____ NEJ

Om ja,varför? _____

_____

_____

4. Använder Ni automatiserade verktyg (t.ex. CASE verktyg) under utvecklingsprocessen?

____ JA

____NEJ

Om ja, hur viktiga är dessa verktyg för att kunna genomföra ett *"lyckat"* mjukvaruprojekt?

____ mycket viktigt

____ viktig

____ inte viktigt alls

____ ingen uppfattning

Utvecklings teamet

5. Hur värderar Du individuell kompetens map de individer som ingår i utvecklingsteamet?

____(Ange en siffra mellan 1-4 där 1 är lägst och 4 är högst)

6a. Hur viktigt är det att bygga rätt utvecklingsteam, d.v.s. att det finns en god balans mellan olika kompetenser, från början?

____(Ange en siffra mellan 1-4 där 1 avser den lägsta värderingen och 4 är den högsta)

6b. Hur viktigt är det att behålla samma teammedlemmarna under hela projektet?

_____ mycket viktigt

_____ viktigt

_____ inte viktigt alls

_____ ingen uppfattning

7. Anser Du det vara viktigt med rollfördelning, d.v.s. att det finns en informationsansvarig, domänexpert, tekniskansvarig o.s.v., i ett utvecklingsprojekt?

____ JA

____ Nej

Om ja,varfor? _____

_____

_____

Projektledare

8. Anger de egenskaper som Du tycker är mest betydelsefull för en projektledare? Markera med 1 den som är mest betydelsefull, 2 för det näst mest betydelsefull och 3 för den som är minst betydelsefull.

( ) erfarenhet

( ) beslutsfattande förmåga

( ) personliga egenskaper

Krav

9. Hur viktigt är det att involvera användaren i kravhanteringsprocessen?

____ mycket viktigt

____ viktigt

____ inte viktigt alls

____ ingen uppfattning

10. Anger de två egenskaperna du tycker är mest betydelsefull vad gäller kravhantering. Markerar med 1 den som är mest betydelsefull och 2 den som är näst mest betydelsefull.

( ) etablera så kompletta krav som möjligt från början

( ) etablera de mest grundläggande krav från början

( ) att ändringar hanteras på ett effektivt sätt

( ) att en specifik metod används för kravhantering

Kunder och användare

11a. Är det viktigt att kunden deltar i hela utvecklingsprocessen?

____ JA

____ NEJ

Omja,varför? _____

_____

_____

11b Är det viktigt att användare deltar i hela utvecklingsprocessen?

_____ JA

_____NEJ

Om ja, varför?_____

_____

Schemaläggning och estimering

12. Hur viktigt är det med verktygstöd för att kunna effektivt estimera t.ex. produkt storlek, ”lines of code”, schemalagd tid o.s.v., i ett mjukvaruprojekt?

____ mycket viktigt

____ viktigt

____ inte viktigt alls

____ ingen uppfattning

13. Anser Du att noggrann schemaläggning, d.v.s. milstolpar, regelbundna rapporteringar o.s.v, är en förutsättning för ett *”lyckat”* mjukvaruprojekt.

____ JA

____ NEJ

Om ja, varför?_____

_____

_____

Övrigt

14. Anser Du det vara viktigt med rutiner, d.v.s. kodinspektioner, ”design reviews”, rutiner för testning o.s.v, för kvalitetskontroll?

___ JA

___ NEJ

Om ja,varför_____

_____

_____

15. Hur viktigt anser Du det vara att ha en väl definierad projektobjektiv?

_____ mycket viktigt

_____ viktigt

_____ inte alls viktigt

_____ ingen uppfattning

2. ATTITYDFRÅGOR

1 Projektledaren borde ha fullmakt för att kunna fatta alla typer av beslut under utvecklingsprocessen.

_____ stämmer _____ stämmer ej

2 Ett mjukvaruprojekt har större chans att lyckas om team medlemmarna känner att de har inflytande över projektet och att de har frihet att skapa en *”god design”*.

_____ stämmer _____ stämmer ej

3 Realistiska ”deadlines” är avgörande för ett projekts framgång.

_____stämmer _____ stämmer ej

4 Bra kommunikations vägar är en förutsättning för ett *”lyckat”* mjukvaruprojekt.

_____ stämmer _____ stämmer ej

5. Ett projekt bör ha en ansvarig person ”ansvarig utgivare” för projektet, d.v.s. någon som ser till att projektet verkligen genomförs?

_____ stämmer _____ stämmer ej

En sista fråga…

Får jag kontakta Er för en kompletterande intervju?

_____ JA _____ NEJ

***TACK** för Er medverkan*

Timothy Little

Frågeguide

Management

- Projekt objektiv, betydelse för framgångsrik projekt
- Ansvarig utgivare (sponsor*)*
- Att lära sig av tidigare misstag. (post mortem analysis)
- Kommunikation
- Noggrann ”Software measurement” (produkt storlek (LOC), schemalagd tid, felaktigheter o.s.v.)
- Rutiner för kvalitetskontroll (”design reviews”, rutiner för testning, kodinspektioner)

Customers and users

- Användarmedverkan (hela utvecklingsprocessen)
- Kundmedverkan (hela utvecklingsprocessen)
- Användaren ha befogenhet att fatta beslut(inte om projektet men map olika lösningar)

The development process

- metodologi vid mjukvaruutveckling,
- automatiserade verktyg (CASE verktyg), vad de bidrar med

The development team

- individuell kompetens
- rätt utvecklingsteam från början
- folk som lämnar projektet
- rollfördelning i utvecklingsteamet (domänexpert, informationsansvarig)
- inflytande över processen (”ownership”)

Requirements

- Användarmedverkan i kravframtagnings processen
- Användaremotstånd(ovilligt att prata om krav)
- Ändringar i krav
- Etablera grundläggande krav eller så kompletta krav som möjligt i början

Project manager

- Egenskaper (erfarenhet, beslutsfattande förmåga, personliga egenskaper)
- Projektledaren ha fullmakt att driva sitt projekt

Scheduling

- Noggrann schemaläggning (milstolpar, regelbundna rapporteringar)
- Realistiska deadlines
- Realistiska estimeringar

Other

Vad tycker du kännetecknar ett lyckat mjukvaruprojekt?