A situational approach and intelligent tool for collaborative requirements elicitation

Chad Raymond Coulin

A Situational Approach and Intelligent Tool for Collaborative Requirements Elicitation

Chad Raymond COULIN

A thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy in Computing Sciences

University of Technology, Sydney

Université Paul Sabatier (Toulouse III)

Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS)

Centre National de la Recherche Scientifique (CNRS)

2007

CERTIFICATE OF AUTHORSHIP/ORIGINALITY

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Candidate

_____

Chad Raymond COULIN

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my principal supervisor, Associate Professor Didar Zowghi, for her wisdom, guidance, and support, throughout my candidature. I would also like to thank my Cotutelle supervisor in France, Professor Abd-El-Kader Sahraoui (LAAS-CNRS and IUT Blagnac), and my co-supervisors Dr Vincenzo Gervasi (University of Pisa) and Dr Yusuf Pisan, for the many discussions, valuable advice, and constant encouragement.

A big thank you goes to my colleagues Norazlin Yusop, Nurmuliani, Dr Samiaji Sarosa, and Dr Mahmood Niazi of the Requirements Engineering group at UTS FIT, and Mohamad Hani El Jamal, Mourad Messaadia, Adel Ouardani, and Nabil Sadou of the Systems Engineering and Integration group at LAAS-CNRS, who made my candidature so enjoyable and rewarding. This thesis would simply not have been possible without the ongoing support and encouragement of my parents Ray and Ruth Coulin, my brother Jarrod Coulin, and my friends Daniel McCormack, Anshu Jayaweera, and Ranga Welaratne. To all of you I am eternally grateful.

A special thanks goes to Professor Alan Davis and Professor Donald Gause for the always insightful and entertaining conversations. And finally, I would like to thank all the people who generously gave their time to participate in the expert interviews, novice questionnaire, case studies, and experiments.

The research presented in this thesis was conducted as part of a Cotutelle agreement between the University of Technology, Sydney and Paul Sabatier University. It was sponsored by the Insearch Doctoral Award Scholarship and the French Embassy in Australia / Australian Academy of Science Cotutelle Grant.

PUBLICATIONS

This thesis includes some parts of the following publications:

Coulin, C & Zowghi, D 2003, 'Agent-Based Support for Requirements Elicitation', ESEC/FSE International Workshop on Intelligent Technologies in Software Engineering, Helsinki, Finland, September 1.

Coulin, C & Zowghi D 2004, 'GONDOLA: An interactive computer game-based teaching and learning environment for Requirements Engineering', Requirements Engineering: Foundation for Software Quality, Riga, Latvia, June 7-8.

Coulin, C 2004, 'A Novel and Collaborative Approach to Requirements Elicitation with Process Guidelines and Intelligent Tool Support', International Conference on Requirements Engineering – Doctoral Consortium, Kyoto, Japan, September 7.

Coulin, C & Zowghi, D 2005, 'Requirements Elicitation for Complex Systems: Theory and Practice', *Requirements Elicitation in Requirements Engineering for Socio-Technical Systems*, Maté, JL & Silva, A (Eds.), Idea Group, USA.

Zowghi, D & Coulin, C 2005, 'Requirements Elicitation: A Survey of Techniques, Approaches, and Tools', *Engineering and Managing Software Requirements*, Aurum, A & Wohlin, C (Eds.), Springer, USA.

Coulin, C, Sahraoui, AEK & Zowghi, D 2005, 'Towards a Collaborative and Combinational Approach to Requirements Elicitation within a Systems Engineering Framework', International Conference on Systems Engineering, Las Vegas, USA, August 16-18.

Coulin, C & Sahraoui, AEK 2005, 'A Meta-Model Based Guided Approach to Collaborative Requirements Elicitation for Software Systems Development', International Conference on Software and Systems Engineering and their Applications, Paris, France, November 29 – December 1.

Coulin, C, Zowghi, D & Sahraoui, AEK 2005, 'A Workshop-Centric Situational Approach for Requirements Elicitation', International Workshop on Situational Requirements Engineering Processes, Paris, France, August 29-30.

Coulin, C & Zowghi, D 2005, 'What Do Experts Think About Elicitation? - A State of Practice Survey', Australian Workshop on Requirements Engineering, Melbourne, Australia, November 22.

Coulin, C, Zowghi, D & Sahraoui, AEK 2006, 'A Situational Method Engineering Approach to Requirements Elicitation Workshops in the Software Development Process', *Software Process: Improvement and Practice*, vol. 11, pp. 451-64.

Table of Contents

CHAPTER 1: Introduction

1.1 Background

Computer systems, and especially software, play an integral part in the lives of most people today. Society continues to grow increasingly more dependent on software, and subsequently their expectations of it continue to rise. Almost twenty years ago it was stated “the hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later” (Brooks 1987). In this respect, very little has changed, as it was more recently suggested “everything else in software development depends on the requirements. If you cannot get stable requirements, you cannot get a predictable plan” (Fowler, M. 2005).

Therefore, the elicitation, analysis, specification, validation, and management of requirements for software systems, collectively known as Requirements Engineering (RE), remains an area of the utmost importance to both researchers and practitioners. Over the past two decades or so, RE has established itself as one of the more critical and challenging activities within the software development lifecycle. In fact more projects for software systems fail as a result of problems with requirements than for any other reason (Hickey & Davis 2003b; The Standish Group 1994, 2003). The complexity of RE in software development, and the wide variety of tasks that it may consist of, has led to separate yet related areas of investigation into the various individual phases that can make up its composition.

The subject of this thesis is requirements elicitation, which can be broadly defined as the process of searching, acquiring, and elaborating the wants and needs of stakeholders for a proposed software system, in terms of requirements, goals, constraints, and features, by means of investigation and exploration. Furthermore, it is generally understood and accepted that requirements are ‘elicited’ rather than just

gathered, captured, or collected. This implies both discovery and development elements of the process. In general, this very social activity can be performed as part of a planning exercise, feasibility study, or larger software development lifecycle, and may consist of numerous tasks performed using a variety of techniques, in order to elicit information from a range of sources.

The research described in this thesis examines, develops, and evaluates a situational approach (i.e. an approach that relates to, and is appropriate for, the situation) to collaborative and combinational requirements elicitation for software systems called OUTSET, and an intelligent support tool with process guidelines named MUSTER. The main objective of this approach and tool is to enable multiple stakeholders to work collaboratively with each other and the analyst, and to use multiple and different requirements elicitation techniques in combination where complementary, all within a workshop environment. The overriding aim of the research is therefore to investigate if such an approach and tool could be designed and constructed, which would lead to improvements in terms of efficiency, effectiveness, and useability. In particular, we were interested in supporting the early stages of requirements elicitation, when conducted by novice analysts, working on projects without a define methodology.

1.2 Problem

Requirements elicitation is a fundamental part of the software development process, but often considered a major problem area in both research and practice, and widely regarded as one of the more challenging activities within the scope of Requirements Engineering (RE). Heavily dependent on the experience and expertise of the participating analyst, the elicitation of requirements is often performed badly in practice, as true experts are few and far between. The subsequent effects of poor software requirements elicitation regularly include costly rework, schedule overruns, poor quality systems, stakeholder dissatisfaction, and project failure (Hickey & Davis 2002). But despite the obvious need for an appropriate level of structure and rigor, this critical, complex, and potentially expensive activity in the majority of cases is performed in an ad-hoc manner without a defined process or methodology.

Compounding this problem is that many of the current techniques, approaches, and tools from research for the elicitation of requirements are either unknown or too complex for novices, and there is a general unwillingness to adopt them in industry, resulting in a significant gap between requirements elicitation theory and practice (Hickey 2003). Just as important is the current gap between expert and novice analysts, which can be attributed to a number of factors, not least of which is the extensive skill set and range of experiences that is often required to successfully conduct this difficult yet vital activity (Hickey & Davis 2003b). A lack of systematic methods with situational process guidance, and experience reports that can easily be applied to real-world situations, are additional reasons for the current state of requirements elicitation in practice.

It is important to note that the elicitation of requirements is an essential part of software development, regardless of the type of project or system under investigation. Furthermore, it is almost inevitable that all Information Technology (IT) professionals will at some point be called upon to conduct requirements elicitation in one form or another. Therefore not only is the production of high quality requirements through effective and efficient elicitation absolutely essential for the engineering of successful software products, but the elicitation of requirements is a fundamental skill for both

current and future IT professionals. As a result, requirements elicitation is an issue of paramount importance in Software Engineering (SE) research and practice, due to the real and continuing need to improve the process, its execution, and the results it produces.

1.3 Scope

Although the subject of requirements elicitation has received some degree of attention in the research literature to date, there still remains a justified need for new approaches and tools that can be easily utilised by the majority of practitioners in typical projects. In this research project we investigated both the state of the art and the state of practice in order to develop and evaluate both an approach and a tool to support the elicitation of requirements for software systems in a workshop environment. The principle focus of this work is on the early stages of requirements elicitation, and not the other RE activities related to analysis and design such as modelling and specification. Furthermore, we concentrated on the fact-finding and information-gathering tasks, and not other tasks often associated with requirements elicitation such as prioritization and negotiation. It is during this initial phase that an appropriate level of structure and rigor would be the most beneficial, given that the requirements are in their most raw form, and the process is at its most fluid.

We have also concentrated on directly addressing the needs of novice analysts working on software development projects without a defined requirements elicitation methodology. This area has been specifically targeted because of its potential to have the most impact, since novices by definition have a low level of expertise with the requirements elicitation, and would therefore benefit from additional process support in the absence of a prescribed methodology. We believe that by addressing the combination of novice analysts, and projects without a detailed and mature requirements elicitation methodology, the research also has a wide coverage in terms of projects, and a large audience in terms of practitioners. Furthermore, we are of the opinion that this subset of software development projects in practice would benefit the most from the adoption of a new and improved approach and tool for requirements elicitation, and especially one that was both collaborative (i.e. enables multiple stakeholders to work together with each other and the analyst) and combinational (i.e. uses multiple and different requirements elicitation techniques where complementary).

Research Assumptions

The research described in this thesis is based on a number of important underlying assumptions, most importantly being that effective requirements elicitation leads to good quality requirements, which in turn leads to high quality systems and more successful projects. However, this assumption is supported by conventional wisdom, often articulated in the available literature (e.g. (Davis, A. M. 1990; Kotonya & Sommerville 1998; Lauesen 2002; Robertson, S. & Robertson 1999). We also presume that the objective of a requirements elicitation process is to elicit all the relevant information from the sources in the most efficient and effective way. Other assumptions made initially, and later supported by the results of our literature review and survey of practice, are that novice analysts actually need more support during requirements elicitation, and to a lesser extent, that approaches and tools are an appropriate way to help narrow the gaps between research and practice, and experts and novices.

1.4 Goals

The general hypothesis of our research is that the process of requirements elicitation for the development of software systems could be improved in terms of efficiency, effectiveness, and useability, by a situational approach and intelligent tool, particularly when performed by novice analysts during the early stages of projects without a defined methodology.

More specifically, we propose that the OUTSET approach and MUSTER tool we have developed, as described and evaluated in this thesis, are not only useful and useable in terms of providing situational and group support, but also improve the efficiency and effectiveness of the requirements elicitation process. Subsequently, the following research goals and related questions were established to examine and validate these hypotheses.

Research Goal 1: Review and critically analyse the existing state of the art in requirements elicitation from the literature, including the existing techniques, approaches, and tools.

Research Question 1: What are the definition, processes, and scope of requirements elicitation for software systems?

Research Question 2: What are the relative strengths and weaknesses of the existing requirements elicitation techniques, approaches, and tools?

Research Question 3: If necessary, what should be the key components of a new and improved approach and tool for requirements elicitation based on the available literature?

Research Goal 2: Investigate and survey the current state of practice in requirements elicitation as reported in the research literature, as well as from the perspective of both expert and novice analysts.

Research Question 4: What does the existing literature say about the state of practice in requirements elicitation?

Research Question 5: What do expert analysts in the field say about the state of practice in requirements elicitation?

Research Question 6: What do novice analysts in the field say about the state of practice in requirements elicitation?

Research Question 7: If necessary, what should be the key components of a new and improved approach and tool for requirements elicitation according to the state of practice?

Research Goal 3: Design an approach and construct a tool to support novice analysts during the early stages of requirements elicitation for software development projects.

Research Question 8: Can the identified key components be combined to design a new and improved approach for requirements elicitation?

Research Question 9: Can the identified key components be combined to construct a tool to support and enhance the new and improved approach?

Research Goal 4: Evaluate the performance of both the approach and tool for requirements elicitation in terms of improvements to efficiency, effectiveness, and useability.

Research Question 10: Does the approach and tool combination improve the efficiency of the requirements elicitation process, in terms of the overall amount of information elicited, when performed by novice analysts?

Research Question 11: Does the approach and tool combination improve the effectiveness of the requirements elicitation process, in terms of the amount of relevant information elicited, when performed by novice analysts?

Research Question 12: Does the approach and tool combination improve the useability of the requirements elicitation process for novice analysts?

1.5 Methodology

From an ontological and epistemological perspective, we have adopted a largely traditional positivist philosophical position for this research (Orlikowski & Baroudi 1991), as opposed to an interpretive or critical view. This is to say that we subscribe to the opinion that there is only one objective reality, which is probabilistic, knowable, and can be described and measured independently of the observer. This does not however preclude us from using both qualitative, i.e. understand the phenomenon of people and the social and cultural contexts within which they live (Myers 2007), and quantitative, i.e. use of values and levels to represent theoretical constructs and concepts in order to understand a phenomenon of people and their context (Straub, Gefen & Boudreau 2005), methods in our research.

Although quantitative and qualitative approaches are often considered fundamentally different, they are not necessarily opposed and there is a growing consensus that both types of research approaches have a great deal to offer (Burrell & Morgan 1979; Long et al. 2000). One of the most common, and arguably simplistic ways in which to differentiate between the two, is in their support of either an objective reality (i.e. quantitative, which relies on statistics and figures), or a subjective reality (i.e. qualitative, which utilizes language and description). The real difference between the two approaches lies in the assumptions made by the researcher, as determined by the context of the study, and the form and focus of the research (Lee 1992).

With respect to the differences between specific qualitative and quantitative methods, this is a matter of degrees, along a continuum between pure objectivism at one end, and total subjectivism at the other (Long et al. 2000; Olson 2006). This suggests that methods of investigation take their qualities from the way in which they are used, and that this subsequently determines their position on the objective-subjective continuum (Long et al. 2000). Accordingly, we consider that specific methods, particularly data gathering, are not necessarily linked with either a qualitative or quantitative approach (Olson 2006).

The benefits of combining multiple research methods, both qualitative and quantitative, are well documented and supported in the literature (see (Myers 2007) for a list of relevant references). Although some have suggested that a mix of qualitative and quantitative methods may cause "ontological oscillation" (Burrell & Morgan 1979), it has also been argued that exclusively qualitative studies often tend to disregard reliability, while exclusively quantitative studies frequently overlook validity. So rather than detracting from either the validity or reliability of the study, the integration of different qualitative and quantitative methods actually enhances the research, as multiple data sources can improve understanding of the context, and provide primary sources of data (e.g. interviews and questionnaires) to support findings from secondary sources such as the literature.

Furthermore, we agree with the position that method selection is primarily a question of appropriateness to the specific problem to be addressed, as determined by the participating researcher (Long et al. 2000; Olson 2006). Consequently, we have combined both qualitative (e.g. interviews and case studies) and quantitative (e.g. questionnaires and formal experiments) methods for our research where most appropriate, based on the available and most suited contextual sources, in order to best address the established research goals and questions. As a result, the research was divided into five sequential but overlapping stages as detailed below.

Stage 1: Literature Review

The first stage of the research was the Literature Review (Coulin & Zowghi 2005a; Zowghi & Coulin 2005), which involved a thorough review and critical analysis of existing theory on and around the area of requirements elicitation for software systems (i.e. state of the art). The primary objective of this review was to establish a preliminary set of approach guidelines and tool features, and to provide a theoretical foundation for the research.

Stage 2: Survey of Practice

The second stage of the research was the Survey of Practice (Coulin & Zowghi 2005b), which involved reviewing the available literature on requirements elicitation in practice. This was followed by in-depth structured interviews with experts, and an online questionnaire for novices, about the current state of practice in requirements

elicitation. The purpose of this survey was two-fold. Firstly, to confirm what exists in the RE literature and what is generally perceived but not proven about the state of practice in requirements elicitation. And secondly, it acted as an elicitation session for the approach guidelines and tool features, thereby practicing what we as requirements researchers preach in terms of the need to conduct thorough requirements elicitation (Davis, A. M. & Hickey 2002).

The triangulation of results from these three methods (i.e. the literature review, the expert interviews, and the novice questionnaire) was critical to the integrity of the research. Asking novices alone is problematic because by definition they typically do not have the variety and depth of experiences to make retrospective comments about the state of practice. Likewise, novices are more likely to be unaware of many of the existing techniques, approaches, and tools that can be used for requirements elicitation. Only asking experts could be similarily problematic, as it may be difficult for them to now imagine the difficulties novices might face, and what is the best type of support needed to overcome them. It is also not possible to depend entirely on the literature due to the work needing to be grounded in practice, and also due to the general lack of empirical evidence and industry-based publications in this area.

Stage 3: Approach Design

The third stage of the research was the Approach Design (Coulin, Zowghi & Sahraoui 2005, 2006), which involved the design of OUTSET, a combinational approach to requirements elicitation based on the principles of Situational Method Engineering (SME). The results from Stage 1 (Literature Review) and 2 (Survey of Practice) of the research provided both motivation and input for the design of the OUTSET approach.

Stage 4: Tool Construction

The fourth state of the research was the Tool Construction, which involved the construction of MUSTER, a collaborative tool to support the OUTSET approach based on the principles of Group Support Systems (GSS). The MUSTER tool therefore represents a proof of concept for the underlying theories and principles used as the basis for the OUTSET approach.

Stage 5: Empirical Evaluations

The fifth and final stage of the research was the Empirical Evaluations, which involved the evaluation of the approach and tool through a case study, case study experiment, and formal experiment (a discussion on the available and selected methods for this stage of the research can be found in Section 6.2). During these evaluations, data was collected and analysed from the elicited information, observation notes, and feedback questionnaires.

Ethical Considerations

For this project Human Research Ethics Committee (HREC) clearance was necessary for both the survey of practice and the empirical evaluations, in accordance with the University of Technology, Sydney (UTS) guidelines. This is to ensure that the research was conducted with honesty and integrity, and the confidentiality and privacy of the participants was maintained where appropriate. In all cases every possible effort was made to minimize the potential risk, and protect the participants from any realistic harm they might experience. This included obtaining formal consent from all the participants using an information letter (see Appendix A) and consent form (see Appendix B), and by de-identifying all the analysed data where it was not possible to collect it anonymously. Although the data was securely stored, no information was recorded that would enable anyone else to identify the research participants individually, such as specific names or characteristics of the respective participants or their actions.

1.6 Contributions

The research presented in this thesis provides a number of valuable **contributions to the body of knowledge** on requirements elicitation including:

1. A detailed review and critical analysis of the current state of the art in requirements elicitation, including the available process models, techniques, approaches, and tools.

2. A detailed survey and primary source of information on the current state of practice in requirements elicitation, including the trends and challenges of expert and novice analysts.

3. A new and improved approach specifically for requirements elicitation workshops called OUTSET, that utilizes, extends, and demonstrates a successful application of Situation Method Engineering (SME) principles.

4. A new and improved tool specifically for requirements elicitation workshops called MUSTER, that utilizes, extends, and demonstrates a successful application of Group Support System (GSS) principles, as well as embodying and enhancing the OUTSET approach.

5. Empirical evidence as to the relative performance of the OUTSET approach and MUSTER tool for requirements elicitation in both theory (experiments) and practice (case study) with respect to efficiency, effectiveness, and useability in particular.

In addition, the research presented in this thesis offers a number of helpful **solutions to the problems in practice** including:

1. Provides practitioners with a new and improved approach and tool combination for requirements elicitation that is flexible, effective, efficient, useable, and useful, which can be readily applied to real-world projects.

2. Encourages and implements an appropriate degree of structure and rigor through process guidance, thereby substituting for a possible deficiency in experience and expertise of the participating novice analyst.

3. Directly addresses two areas of requirements elicitation workshops that were identified as lacking and needed, namely situational process guidance and group interaction support.

4. Directly addresses some of the issues in requirements elicitation that can lead to software project failure, by enabling stakeholder input early in the development process, and by encouraging a structured and rigorous approach.

5. By using the presented approach and tool, analysts will be better able to manage and reference large amounts of information, and have greater control over the process and the consistency of results.

Novelty and Originality

The novelty and originality of the presented research was not restricted to the specific context that was investigated (i.e. the support of novice analysts during the early stages of requirements elicitation for projects), or the particular methodology that was used (i.e. the triangulation of methods for the development of the OUTSET approach and MUSTER tool). The use of Situational Method Engineering (SME) principles as the basis for the approach, and Group Support System (GSS) functionalities combined with CAME/CASE characteristics for the tool, was unique from any other previous work on requirements elicitation. Furthermore, the use of only Open Source technologies, and a plug-in architecture as a mechanism to store and transfer expert requirements elicitation knowledge was not only new and innovative, but also allowed us to implement intelligence into the tool for process guidance and cognitive support.

1.7 Outline

This thesis for the research project described is structured as follows:

Chapter 1: Introduction

In this chapter we introduce the research, including the problem, scope, goals, methodology, and contributions. The purpose of the chapter is therefore to lay the foundation for the research and overview the rest of the thesis.

Chapter 2: Literature Review

In this chapter we review the available literature on and around the subject of requirements elicitation, including the existing techniques, approaches, and tools. The purpose of the chapter is therefore to provide the theoretical basis for the research by critically analysing the current state of the art in requirements elicitation.

Chapter 3: Survey of Practice

In this chapter we survey the current state of practice in requirements elicitation, by presenting the results of in-depth interviews we conducted with experts, and an online questionnaire for novices. The purpose of this chapter is therefore to identify the key trends of experts, and major challenges for novices, when performing requirements elicitation.

Chapter 4: The OUTSET Approach

In this chapter we present an approach to the early stages of requirements elicitation based on Situational Method Engineering (SME). The purpose of this chapter is therefore to provide novice practitioners with a new and improved approach to requirements elicitation that can be easily applied to real-world projects.

Chapter 5: The MUSTER Tool

In this chapter we present a Group Support System (GSS) to enhance and extend the OUTSET approach. The purpose of this chapter is therefore to provide novice practitioners with a tool that makes the collaborative and combinational OUTSET approach more useable and useful for requirements elicitation workshops.

Chapter 6: Empirical Evaluations

In this chapter we empirically evaluate a specific instance of the OUTSET approach embodied in the MUSTER tool, by way of a case study, case study experiment, and formal experiment. The purpose of this chapter is therefore to determine the relative useability, efficiency, and effectiveness of our approach and tool combination.

Chapter 7: Conclusions

In this chapter we conclude the research by detailing the contributions to the body of knowledge, the solutions to the problems in practice, and the novelty of both the research and the results. The purpose of this chapter is therefore to summarize the importance, value, and originality of the thesis.

This thesis also contains the following **Appendices**:

Appendix A: Invitation Letter to participate in the research

Appendix B: Consent Form for participation in the research

Appendix C: Expert interview questions

Appendix D: Online novice questionnaire

Appendix E: Requirements Specification for the MUSTER tool

Appendix F: Feedback Questionnaire used in the evaluations

Appendix G: Observation Sheet used in the evaluations

CHAPTER 2: A Review of Theory

2.1 Chapter Overview

In this chapter we will review the relevant literature on and around the topic of requirements elicitation as it relates to our research. We will begin by defining requirements elicitation with respect to software development projects (Section 2.2), and then describe the fundamental activities of the requirements elicitation process (Section 2.3). The next three sections will constitute a critical analysis of the state of the art in requirements elicitation focusing on the available techniques (Section 2.4), approaches (Section 2.5), and tools (Section 2.6). This will be followed by a summary of the entire chapter (Section 2.7).

The purpose of this chapter is therefore to provide a theoretical foundation for our research by investigating the state of the art in requirements elicitation (**Research Goal 1**). We aim not only to define the process and scope of requirements elicitation for software systems (*Research Question 1*), but to also identify the relative strengths and weaknesses of the existing requirements elicitation techniques, approaches, and tools (*Research Question 2*). The results from this review of theory will be combined with the results from our survey of requirements elicitation practice (Chapter 3) to form the basis of both the key components of the OUTSET approach presented in Chapter 4, and the key features of the MUSTER tool detailed in Chapter 5 (*Research Question 3*).

2.2 Requirements Elicitation

2.2.1 Section Overview

In this section we will further introduce the area of investigation by defining and describing what is meant by 'requirement', 'Requirements Engineering', and specifically 'requirements elicitation', in relation to Software Engineering (SE). The purpose of this section is therefore to define in more detail the key concepts in our research, and the main topic we have addressed, and the position and relevance of requirements elicitation to system development. We consequently lay the foundation for the rest of the chapter by clarifying the specific area of research we are concerned with, and place it within the activities of software development projects.

2.2.2 What is a 'requirement'?

Before we investigate Requirements Engineering, and subsequently requirements elicitation, it is prudent to first define the word that is common in both of these terms, and what it means within the context of software engineering. The IEEE (IEEE 1990) defines a requirement as "(1) A condition or capability needed by a user to solve a problem or achieve an objective, (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents, or (3) A documented representation of a condition or capability as in 1 or 2". Put another way, a requirement is a statement of a customer need or objective, or of a condition or capability that a product must possess to satisfy such a need or objective, or a property that a product must have to provide value to a stakeholder (Wiegers 2007).

In more practical terms, these definitions correctly identify a requirement as something that the system must do, must have, or must satisfy, as determined by someone related to its development. Al Davis (Davis, A. M. 2004) takes it one step further by stating that a requirement must be an "externally visible characteristic" of the desired system. We can see from this variety of definitions, that there appears to

be a lack of general agreement in research as to what is a requirement, and although implied to some degree but missing from all of these is a reference to requirements as the criteria for judging the success of the system development effort and therefore the success of software projects (George Mason University 2005).

Requirements are not only used as the basis for other downstream activities in the software development lifecycle such as design, testing, and system acceptance, but also in the planning phase to estimate the development cost and schedule for the system, as well to determine the feasibility and value of the project. Along these lines Maiden and Rugg (Maiden, N. A. M. & Rugg 1996) state that requirements are used for three basic purposes being "(1) to provide a specification for the design and implementation of a system, (2) to act as criteria for the selection of a system package, and (3) to form the basis of procurement agreements such as legal contracts between suppliers and customers". Consequently, requirements play a pivotal role in both system acquisition and system development types of projects.

There can be many different types of requirements for software systems, and just as many ways to categorizing and classify them. The most common way is by differentiating between Functional and Non-functional requirements. Robertson and Robertson (Robertson, S. & Robertson 1999) state that functional requirements are things the product must do, whereas non-functional requirements are qualities the product must have. Along the same lines it can be said that functional requirements represents features and capabilities of the system, where as non-functional requirements which are equally important include those relating to thing such as useability, reliability, maintainability, and other software quality attributes of the target system (Sommerville 2001). The types of requirement can be further decomposed into sub-types in order to form a hierarchy of requirements statements. For example, a Non-Functional Requirement can be a Performance Requirement, a Quality Attribute, or a Design Constraint (Chung et al. 1999). Another common method of classifying requirements is by grouping them as System, User, Business or Domain related (Sommerville 2001).

2.2.3 What is 'Requirements Engineering'?

Software Engineering (SE), as a branch of larger Systems Engineering discipline, is the area of Computer Science concerned with the development of software-intensive computer-based systems, as well as the processes, methods, and tools used to accomplish it (Finkelstein & Kramer 2000). Software is typically engineered within a project and development lifecycle, where the main Requirements Engineering (RE) effort is performed after project initiation, but before system design. This is traditionally followed by coding, testing, operation, and maintenance phases as can be seen in Figure 2.2.1. However RE can also be performed iteratively and incrementally throughout the software development lifecycle, and the results of the RE stage can also be used for planning purposes and to determine if a project should continue at all (i.e. a feasibility study).

It is important to note that in the most general terms, RE is concerned with understanding what a system must do (the 'what'), where as the design phase is concerned with how it should do it (the 'how'). An often-cited definition by Zave (Zave 1997) states "Requirements Engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems". Sommerville and Sawyer (Sommerville & Sawyer 1997) go further and describe RE as "the activities that cover discovering, analysing, documenting and maintaining a set of requirements for a system". Both of these definitions rightfully suggest that RE is more than just the collection of facts, but that it encompasses all the project lifecycle activities associated with understanding the necessary capabilities and attributes of a system (Wiegers 2007).

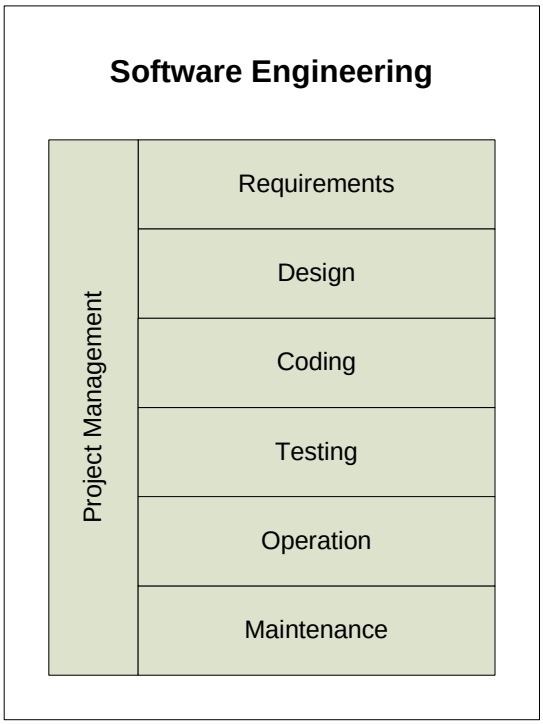

Figure 2.2.1: The phases of Software Engineering (SE)

The actual term ‘Requirements Engineering’ or ‘RE’ as it is commonly known, really started to be used in the late 1970’s within Software Engineering conferences and publications, although in was still more commonly referred to as either ‘Requirements Management’ or ‘Requirements Analysis’ (Buren & Cook 1998). However the RE name and field was only formally established in the early 1990’s, primary as a result of the first conference entirely dedicated to requirements related topics, being the International Symposium on Requirements Engineering in 1993 (RE’93). The community that evolved as a result of this conference, and others like it, accordingly concerned itself with the discovery, analysis, presentation, and management of system requirements.

With its origins in Software Engineering and in particular systems analysis, RE was presented as the fundamental first step of any system development project. As a result RE was positioned as being primarily important for both feasibility studies and system specification processes. Much like the phases of Software Engineering itself, the individual tasks and activities within the RE process are commonly divided into a number of phases. Once again there are a number of ways to group these activities together, and several different definitions for each of these resultant phases. However

a typical RE process includes the phases of Elicitation, Analysis, Specification, Validation, and Management (Kotonya & Sommerville 1998) as described below and shown in Figure 2.2.2.

- Requirements **Elicitation** (which is the core topic of our research, and described in detail in the next subsection) is concerned with the collection, capture, discovery, and development of requirements from a variety of sources including human stakeholders.

- Requirements **Analysis** focuses on examining, understanding, and modelling the elicited requirements, and checking them for quality in terms of correctness, completeness, clarity, and consistency.

- Requirements **Specification** is the act of recording and documenting the requirements in a way that can be used by the stakeholders, and especially the developers who will design and construct the system.

- Requirements **Validation** is the process of confirming the quality of the requirements, and ensuring that they actually represent the wants and needs of the stakeholders.

- Requirements **Management** is performed throughout RE process and includes activities such as change control, version control, requirements status tracking, and requirements tracing.

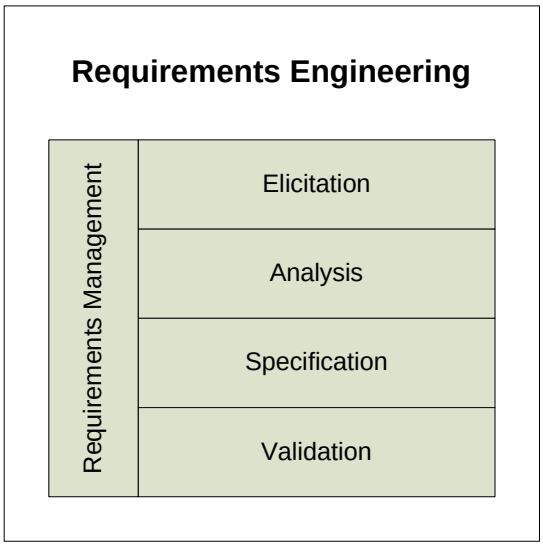

Figure 2.2.2: The phases of Requirements Engineering (RE)

Several variations exist to this set of phases, such as those proposed by (Thayer & Dorfman 1987) and (Sommerville & Sawyer 1997). Others have included the addition phases of negotiation and prioritisation, while Hickey and Davis (Hickey & Davis 2003c) introduced a phase known as Requirements Triage, where the subset of all the requirements elicited that will actually be used is decided. In other breakdowns these additional phases may be included as activities in either the elicitation or analysis phases. In reality the phases of RE are intertwined and performed in an interleaved manner, often in parallel rather than as individual and sequential phases (Hickey & Davis 2003c). This is particularly the case with elicitation and analysis. Requirements elicitation precedes requirements analysis but they are closely interrelated and typically performed iteratively. Furthermore, many of the activities performed during RE are often performed across more than one of these phases. For example, the activity of removing irrelevant and design-related information (the 'how' rather than the 'what') is often performed incrementally as part of the elicitation, analysis, specification, and validation phases.

2.2.4 What is 'Requirements Elicitation'?

As described in the previous subsection, requirements elicitation is typically the first phase of RE. It can therefore be argued that requirements elicitation is in fact the first stage of the software development lifecycle, and consequently a prerequisite for all

the other major development activities. In fact Moore and Shipman suggest that the elicitation of "meaningful requirements from end users is an early and major goal of any and all software engineering processes" (Moore & Shipman 2000). Despite this, the term 'requirements elicitation' is relatively new even within the RE literature, and is still often referred to as requirements collection, capture, acquisition, determination, gathering, identification, invention, development, discovery, and fact-finding. For one reason or another these other terms have been insufficient in representing the real activities, and subsequently it is now generally understood and accepted that requirements are elicited rather than just captured or collected (Goguen, J.A. 1996). This implies that there are discovery, emergence, and development elements to the elicitation process.

Currently there is very little uniformity in RE research and practice concerning a standard definition for requirements elicitation. Hickey and Davis (Hickey & Davis 2003c) define requirements elicitation as "learning, uncovering, extracting, surfacing, and/or discovering needs of customers, users, and other potential stakeholders". An alternative definition offered is that requirements elicitation is "the process of identifying software or system requirements from various sources through interviews, workshops, workflow and task analysis, document analysis, and other mechanisms" (Wiegers 2007). We can elaborate further on these definitions by saying that requirements elicitation is all about learning and understanding the needs, desires, and expectations of users and customers, with the ultimate aim of communicating these to all the stakeholders and especially the system developers. For the most part, however, requirements elicitation is dedicated to uncovering, extracting, acquiring, and elaborating the wants of human stakeholders. Robertson and Robertson (Robertson, S. & Robertson 1999) refer to this process as "trawling for requirements" to highlight the fact that through this process you are likely to get more requirements than expected, and imply that gathering a few extraneous requirements initially is always better than gathering less.

As previously mentioned, requirements elicitation is closely interrelated to the other RE activities, and in particular the phase of analysis. In fact some argue that elicitation is in itself a form of modelling, since during elicitation the analyst creates informal mental models as the problem and the needs become more apparent

(Loucopoulos & Champion 1990). Loucopoulos and Champion argue that requirements elicitation (referred to by them as 'concept acquisition') "is not so much a translation process of a user wish list as an interactive problem-solving process" and further state that "the very early stages of requirements capture (namely the elicitation of concepts about the application domain and the analysis of these concepts) need to be supported by an approach which permits the construction of informal models and the use of scenarios about the modelled phenomena". Although we acknowledge that requirements elicitation and subsequently domain analysis is certainly part of problem solving, imposing the use of actual models or a formal modelling technique at this early stage in the software development process not only requires significant expertise by the analyst, but also can restrict and confine the rich source of information requirements elicitation can produce more informally.

Over the years requirements elicitation research has continued to become more and more multidisciplinary. This has been a result of a need to address not only the technical and functional aspects of the system, but also the human and social factors. Apart from the obvious inheritance from Computer Science, Software Engineering, and naturally RE, we will see later in this chapter that many of the techniques, approaches, and tools used for requirements elicitation come from areas such as Knowledge Engineering and Artificial Intelligence. We will also see that requirements elicitation research has borrowed successfully from many of the softer sciences such as Cognitive Psychology, in order to understand the needs of people and their difficulties in understanding those needs, and Anthropology to determine how people interact with systems and the effect on the environment and their tasks. Sociology and Linguistics also play an important role in requirements elicitation research, as the process is both person-orientated and communication-rich.

2.2.5 Section Summary

In terms of Software Engineering, we have defined a 'requirement' as a statement of a need or objective for a proposed system, or a condition or capability that a product must possess to satisfy such a need or objective. We have identified that requirements are very important to system development, and are used throughout a software

development project, from the planning phase all the way through to system acceptance. It was established that Requirements Engineering or 'RE' is often the first stage in the software development lifecycle, and that it is typically composed of several phases, where the requirements are first elicited from stakeholders and other sources, modelled and analysed for quality, then documented and validated by the stakeholders to produce a set of requirements. The objective of RE is therefore to efficiently and effectively produce and manage a high quality set of requirements that not only represent the real wants of the stakeholders, but also fulfil their intended purpose, whether that be the basis for a design, or the foundation for a contract.

The main topic of our research being 'requirements elicitation' was broadly defined as the earliest phase of software development, closely related to but preceding analysis and design, where the goals, functions, and constraints of a system and the needs, desires, and expectations of the users and customers are uncovered and determined by various means of investigation and analysis. But it is also about understanding these requirements, and involves listening to and learning from stakeholders, providing stimulus to discover and develop more requirements, and enabling externalization of the information. We identified that requirements elicitation is multidisciplinary and as a result research in this area often walks the blurry line between the soft and hard sciences. We will see in detail that requirements elicitation is a complex process (see Section 2.3) that is performed based on a foundation of various and different techniques (see Section 2.4), approaches (see Section 2.5), and tools (see Section 2.6).

In summary, the key points identified in this section are:

1. Requirements are important to, and used throughout, software development

2. Requirements Engineering (RE) is typically the first phase of Software Engineering, and Requirements elicitation is normally the first phase of RE

3. Requirements elicitation is closely related to requirements analysis

4. Requirements elicitation research is multidisciplinary

2.3 The Requirements Elicitation Process

2.3.1 Section Overview

In this section we will follow on from the definition of requirements elicitation in the previous section, by expanding on the actual process itself. A process can be defined as a set of interrelated work activities characterized by a set of specific inputs and value added tasks that make up a procedure for a set of specific outputs performed for a given purpose (American Society for Quality 2006). We will look at the different contexts in which the process of requirements elicitation may be performed, and then investigate the various and relevant process models proposed in the literature. We will then examine and discuss the major activities that make up this complicated and critical process. The purpose of this section is to describe in detail the process in terms of the inputs, activities, tasks, and outputs relating to the elicitation of requirements for software systems. The aim is therefore to provide an overall picture of the requirements elicitation process as the foundation for investigating the techniques, approaches, and tools used for its execution in the subsequent sections.

2.3.2 Elicitation Contexts

Requirements elicitation does not occur in a vacuum, but can be performed in a wide variety of contexts (Hickey & Davis 2003b), and is strongly related to those contexts in which it is conducted. Requirements elicitation can be part of the initiation, planning, or analysis phases of software development, or as part of a larger business process reengineering project. Sometimes the requirements elicitation phase is used as part of a decision making process to determine if a proposed project is viable, desirable, and should go ahead. Hickey and Davis (Hickey & Davis 2003b) have identified a number of different settings representing different project types in which requirements elicitation can be performed. These include the development of customized systems for specific customers (bespoke), the development of commercial-off-the-shelf products for the common market (COTS), the identification, evaluation, and selection of alternative solutions for procurement purposes via a

Request for Proposals (RFP), and the implementation (configuration and customisation) of large and complex systems. Other examples include replacement or conversion of existing systems, and software maintenance and upgrade projects.

In the most common case of custom software development, a number of other alternatives may significantly shape the context, such as whether the organisation intends to build the system internally, buy the system from an external party, or build the system using an external party on contract or time and material (Satzinger, Jackson & Burd 2002). Requirements elicitation may in fact be part of a project or process to determine which of these options the organisation selects. Furthermore there are many kinds of systems that may be developed including distributed, web-based, and embedded just to name a few. The possible permutations of these situational characteristics are numerous. Moreover, there are number of other internal and external factors that may affect the project and how it is conducted including government regulations, changing market conditions, political considerations within the organisation, and the technical maturity of the organisation and users of the target system. Therefore clearly requirements elicitation is highly dependent on specific project, organisational, and environmental characteristics (Christel & Kang 1992).

2.3.3 Process Models

Despite the large array of possible contexts described above, a number of process models have been proposed for requirements elicitation over the years (Christel & Kang 1992; Constantine & Lockwood 1999; Kotonya & Sommerville 1998; Pohl 1996; Robertson, S. & Robertson 1999; Sommerville & Sawyer 1997) based around different variations of a core set of basic activities. For the most part these models have provided only a generic roadmap of the process with sufficient flexibility to accommodate the key contextual differences of individual projects. The inability of these models to provide definitive guidelines is a result of the wide range of tasks that may be performed during requirements elicitation, and the sequence of those activities being dependent on specific project circumstances. The variety of issues that may be faced and the number of techniques, approaches, and tools available to be used only makes the task of developing a useful and useable process model more complex.

Furthermore in most cases the process of requirements elicitation is performed incrementally over multiple sessions, iteratively to increasing levels of detail, and at least partially in parallel with other system development activities (OPEN Process Framework 2007). In order to identify and elaborate on the shortcomings of existing process models, it is useful to determine the core components of the requirements elicitation process to guide the development of more detailed yet flexible models.

Sommerville and Sawyer (Sommerville & Sawyer 1997) state that the requirements elicitation process involves the understanding of "1) the problem to be solved, 2) the business process of the organisation, 3) the way in which the system is to be used, and 4) the application domain of the system". Following along the same lines Kotonya and Sommerville (Kotonya & Sommerville 1998) suggest that a good requirements elicitation process will include "1) object setting (goals, problems, budget, schedule, constraints), 2) background knowledge acquisition (organisation and structure, application domain, existing systems), 3) knowledge organisation (identify stakeholders, roles and responsibilities), and finally 4) stakeholder requirements collection (user, domain and organisational requirements)". Wiegers (Wiegers 2003) takes this one step further by suggesting that an elicitation plan should be developed, which should include among other things, elicitation objectives, elicitation strategies and processes, products of elicitation efforts, schedule and resource estimates, and elicitation risks.

Although these all address the fundamental makeup of a requirements elicitation process, they provide little more than a list of the very high-level areas of required investigation. In response to this situation, a number of attempts have been made to provide more specific and instructive guidelines for the process of requirements elicitation. One such attempt was by Hickey and Davis (Hickey & Davis 2003c, 2004) where they proposed a unified model for the requirements elicitation process using mathematical principles. Although this did provide researchers with a more structured and complete picture of the process, it offered little in the way of hands on guidance for analysts. Andreou (Andreou 2003) on the other hand proposed a step-by-step methodology using the ISO 9126 quality characteristics as a guiding component for the analyst to collect human, social, and organisational factors that can enhance the quality of the software being developed. Although this largely did not take advantage

or integrate much of the existing requirements elicitation research to date, it did, however, provide practitioners with a more tangible and detailed set of instructions for performing requirements elicitation yet at a fairly high level of abstraction.

One of the reasons for this gap in requirements elicitation process guidelines as we have previously mentioned is the large number of situational variables and elements that can affect how the process should and can be performed. One of the more obvious of these is the type of software development process model that is employed, and which the RE and therefore the requirements elicitation activities must support. The type of software development model chosen, such as the Waterfall (Royce 1970), V (Rook 1986), Spiral (Boehm, B. W. 1988), Evolutionary (Gilb 1988), or Incremental (Basili & Turner 1975) models each have their own respective model for RE, which in turn effects the specific requirements elicitation process. An example of this can be seen in Kotonya and Sommerville presentation of the Spiral model for RE (Kotonya & Sommerville 1998).

Therefore, what is needed to improve our understanding of requirements elicitation is a more detailed investigating into the common and underlying activities of typical requirements elicitation processes. To this end and to present our own overview of the requirements elicitation process, as once again there is very little uniformity in the research literature and practice concerning the names given to the activities often performed during requirements elicitation. Subsequently, we have divided the various individual requirements elicitation tasks into five fundamental and interrelated activities as listed below and described in the following subsections. The five requirements elicitation activities described are:

1. Understanding the Domains
2. Identifying the Sources
3. Selecting the Methods
4. Eliciting the Requirements
5. Organizing the Information

2.3.4 Understanding the Domains

The process of requirements elicitation can have a variety of starting points such as a need, an opportunity, or an idea (Gause & Weinberg 1989), however, more typically it is a problem that must be solved. This may be the result of dissatisfaction with the current situation, the development of a new situation, or an opportunity to improve the current situation in terms of time and cost. Typically the process begins with an informal and incomplete high-level mission statement for the project (Zowghi 1999). This may be represented by a set of fundamental goals, functions, and constraints for the target system, or as an explanation of the problems to be solved. In order to develop this description, stakeholders and other sources are identified and used to elicit information about the environment in which the system will be situated. These preliminary results form the basis of further investigation and refinement of this information in a typically iterative and incremental manner.

It is important at the beginning of the requirements elicitation process to investigate and examine methodically and in detail the situation or real-world in which the system will ultimately reside, sometimes called the 'problem domain' (Zave & Jackson 1997), in addition to the type of system being developed, often referred to as the 'application domain' (Jackson 1995). These can be collectively referred to as the operating environment of the system, and needs to be thoroughly explored in addition to the technical, political, organisational, and social aspects related to the project and the system, as well as any constraints that may need to be enforced. Existing work processes and the related problems to be solved by the system need to be described with respect to the key business goals and issues. This is sometimes referred to as 'bounding' as it is the task of determining the scope and environment within which the detailed investigation should take place. A recent study (Chatzoglou 1997) found that "projects with a moderately or well defined structure of the problem domain usually performed less iterations" of the requirements elicitation process. Gause and Weinberg (Gause & Weinberg 1989) suggest that this important first step can be best achieved by asking some context-free questions (i.e. high-level and generically appropriate) to the key stakeholders.

An investigation of both the problem and application domains involves understanding the different elements of the entire context. This includes not only the type of project to be performed and the type of system to be developed, but also the size and kind of the organisation, and the sort of business it conducts (Yeates & Wakefield 2004). In fact there are so many contextual elements that could possibly affect the project and system that it would be difficult to list them all out, but some of the more important internal factors not already mentioned include the technical maturity of the organisation and users, the culture and politics within the organisation, and the existing environment with respect to current systems and processes. External factors such as the economy, the specific market, related laws and regulations, and the existence of outside stakeholders also can have a bearing on how the process should be performed. In practice, the budget, schedule, and availability of resources, within the project can also have a significant impact on the target system. Evidently, it is important to identify and examine the potentially large and diverse number of internal and external contextual factors related to the domains, which may have an effect on how the process of requirements elicitation can and should be conducted (Shelly, Cashman & Rosenblatt 2003).

2.3.5 Identifying the Sources

In most cases the requirements for a software system will be spread across, and will need to be elicited from, a number of sources in a variety of formats (Loucopoulos & Karakostas 1995). The current systems and processes already in place represent a primary source for eliciting requirements, particularly when the project involves replacing an existing legacy system. These may include both upstream and downstream activities of the tasks to be actually supported by the new system. Documentation on the current systems, processes, organisation, and environment can also provide a detailed foundation of requirements information in addition to its supporting rationale and relative importance (Rayson, Garside & Sawyer 2000). This could include manuals, forms, report, company and competitor literature, standards, regulations, and artefacts from other projects conducted within the same organisation such as specifications and designs.

Stakeholders, and more specifically the eventual end-users of the system, represent the most obvious source of determining requirements. One of the first steps in requirements elicitation therefore is to identify and analyse all the relevant stakeholders (Robertson, S. & Robertson 1999). This includes discovery of their individual goals, ideas, motives, and what potential control or influence they may have in the project (Gottesdiener 2002). All parties involved and affected, actively or passively, directly or indirectly, by the development and implementation of the target system should be considered as stakeholders (Sharp, Finkelstein & Galal 1999). Therefore stakeholders can include the client, customers, management, users, subject matter experts, project members, developers, other departments, system administrators and maintenance personnel, training staff, help desk and hotline staff, the organisation's customers, partners, and competitors, professional bodies, authorities, and other special interest groups (see (Bostrum 1989), (Robertson, S. & Robertson 1999), and (Alexander, I. F. 2005) for extensive lists of potential project stakeholders and requirements sources that should be consulted).

More specifically, users can be the source of information about problems with the existing systems, and suggestions for improvements to future systems, especially with respect to business process alignment and useability. Different types of user classes can also be distinguished depending on a number of factors (Wiegers 2003), including their frequency of use of the system, their experience with computer systems, the features they use, the tasks they perform, and the access privileges or security levels they are entitled to. The customer and management are typically responsible for setting the high-level goals and constraints for the system, subject matter experts (also called domain experts) are consulted to obtain detailed information about the problem and solution domains and the environment of the system, and developers and technology experts are used to determine if something is possible and how technology can be used and taken advantage of to solve the problems. It is therefore critical for successful requirements elicitation that all the target system stakeholders are involved or at least considered in the process from an early stage (Wiegers 2006). Although the level of input stakeholders will have on the project and the system will depend ultimately on their availability, cooperation, and ownership, most researchers and practitioners agree that user involvement in particular is critical to the success of the requirements elicitation process (Hickey, Dean & Nunamaker 1999).

2.3.6 Selecting the Methods

Once the sources of requirements and the specific stakeholders have been identified, the actual task of eliciting the core requirements begins with the selection of the techniques, approaches, and tools to be used. Although some may advocate that just one elicitation technique or a single approach is sufficient and may be applied to all cases, it is generally accepted that an individual requirements elicitation technique or approach cannot possibly be suitable for all projects and situations (Hickey & Davis 2003b, 2003c). In fact the quality of the requirements is greatly influenced by the methods selected and employed during elicitation (Hickey & Davis 2002). Requirements elicitation is such a complex process involving many activities that it is typically best performed by using a range and variety of the available techniques, approaches, and tools. By using a combination of complimentary elicitation methods, many of the issues commonly associated with this process can be avoided or at least minimized. The relative strengths and weaknesses of these methods determine when each is appropriate depending on the context and situation. The choice of methods to be used is therefore strongly related to the specific context of the project, organisation, and environment (Christel & Kang 1992). Despite it often being a critical factor in the success of the elicitation process, technique selection has received relatively little attention (Nuseibeh & Easterbrook 2000), however, there are some notable exceptions as described below.

Maiden and Rugg in their ACRE (ACquisition of REquirements) framework (Maiden, N. A. M. & Rugg 1996) provide descriptions, preconditions for use, perceived strengths and weaknesses, and useful references for twelve acquisition methods (all of which are covered in Section 2.4 of this chapter). Their six facet criteria to determine method selection is based on the specific context of the requirements elicitation process, and in particular 1) the purpose of the project, 2) the types of knowledge that can be acquired from stakeholders, 3) the types of knowledge that can be observed in the domain, 3) the problems that are likely when attempting to acquire this knowledge, 4) the stakeholders that should be observed and communicated with, and 6) the practical constraints on each requirements elicitation session. Hickey and Davis

(Hickey & Davis 2002, 2003b, 2003c) have also investigated elicitation technique selection at length, and state that a particular elicitation technique may be selected for any combination of four reasons being a) it is the only technique that the analyst knows, b) it is the analyst's favourite technique for all situations, c) the analyst is following some prescribed or explicit methodology that is being followed for the system development, and d) the analyst understands that the technique is effective in the current circumstance. Furthermore, they argue that most practicing analysts do not have the necessary insight for d), and so depend on one of the first three. Both of these efforts have attempted to develop and validate a set of tentative relationships between the characteristics of a project and the methods to be used as a guideline for selecting technique combinations. At a minimum, they advocate that consideration should be given to the types of stakeholders involved in the process, the information that needs to be elicited, and the stage of elicitation efforts in the project.

A sentiment shared by several including (Maiden, N. A. M. & Rugg 1996) and (Hickey & Davis 2003c) is that technique selection is not fixed, in that during a requirements elicitation process multiple techniques need to be selected and used, and the selection of the next technique will be dependent on the results from the previous efforts. Therefore technique selection itself is iterative. Hickey and Davis state that "the right technique to apply in a given situation must be a function of what requirements we already know and what requirements we still need to know; after all, different techniques are good at uncovering different kinds of requirements" (Hickey & Davis 2003c). Requirements elicitation technique selection depends on the experience and expertise of the analysts and the participants, the specific problem and situation, and what other techniques can and will be used. It is also dependent on a large number of factors including the type of system being developed, the stage of the project, the application domain, time, cost, and the availability of resources to name only a few.

2.3.7 Eliciting the Requirements

During the actual task of eliciting the requirements, it is important to establish the level of scope for the system and investigate in detail the needs and wants of the

stakeholders, typically over a number of sessions using a range of selected methods. It is also essential to determine the future processes the system will perform with respect to the business operations, and examine the ways in which the system may support them in order to satisfy the major objectives and address the key problems of the business. Although the process is called requirements elicitation, in reality more than just the requirements for the system are elicited. The process is also concerned with gathering other types of information on the organisation, the people, and the environment including system goals, business rules, work processes, assumptions, constraints and implementation details. Lauesen (Lauesen 2002) stresses how important it is to elicit other types of information during requirements elicitation in order to complete the picture, including a description of the present work in the domain, a list of the present problems, goals and critical issues, ideas and risk, among others. This is consistent with the perspective of concept modelling, where information is elicited about the processes, the data, and the behavioural aspects of the target system (Rolland & Prakash 2000), and also supported by the traditional structured analysis and design view of software development (Shelly, Cashman & Rosenblatt 2003) where there is a need to collect information on not just the system, but the hardware, software, data, processes and people as well.

Requirements alone offer only very limited value to the stakeholders and other participants of the subsequent phases of the software development life cycle (especially the developers). This information must be associated to a part of the target system (such as a feature, object, or agent) where possible to have any real meaning, and needs to be supported and linked to additional contextual information such as goals, scenarios, and domain facts. Goals are typically the major and high-level business drivers for the project and are things such as increase productivity, and reduce running costs. Scenarios represent the work tasks that the system must support during operation, and domain facts provide information regarding the operating environment of the system and the users. Determining the priority, source, and particularly the rationale of the requirements and goals are also very important elements of the requirements elicitation process (Gambhir 2001). This supports the view that requirements elicitation is the discovery of not only what is needed, but also why it is needed (Graham 1998). Therefore during requirements elicitation, it is essential to elicit more than just the system requirements.

2.3.8 Organizing the Information

Once the required information has been elicited from all the available sources, it then needs to be organized and integrated in such a way as can be later analysed, specified, and validated, as well as negotiated and prioritized. This process involves merging the requirements from the different sources, categorizing them, and storing them in a standardized format. Typically the deliverable from the requirements elicitation phase, and eventually the entire RE process, is a natural language text document or a list of candidate requirements containing different types of information including goals, constraints, requirements, and scenarios, with high-level descriptions of the solution system (Hickey & Davis 2003b). This is often referred to as a Software Requirements Specification or SRS. The output of the requirements elicitation process however is also ultimately dependent on the intended audience and therefore needs to be in a format understandable to those people. In whatever form it takes ideally it should say what the problem is and what is needed to address that problem. Apart from a general understanding of what the system should do, there are also desired but intangible outputs of an elicitation process such as the setting of realistic stakeholders and project expectations, and ensuring that the participants are satisfied, motivated, and committed to the rest of the project. In general a successful requirements elicitation process will produce a quality requirements document that reflects a common vision and provides a better understanding of the problems, constraints, needs, and wants related to the development of the system.

Therefore it is important to achieve the right level of detail for the requirements information. A too high level may make the requirements ambiguous, where as too detailed may unnecessarily restrict the available solution and design options. The different types of information elicited from the different sources may be at different levels of detail. This is normal and just one of the many challenges of RE and specifically elicitation. With respect to the formality of the requirements, natural language (textual), tabular, or simple diagrammatic forms are the most common and appropriate due to the type of participants typically involved during elicitation, and the type of information normally desired at this point in the development process.

Even for safety critical systems, the initial requirements are often expressed in terms of natural language as this is the only notation common to all stakeholders. Therefore the level of required formality at this early stage is typically low.

Although we readily acknowledge the value and usefulness of semiformal and formal methods and notations in the larger RE process (especially for safety critical systems), we believe that these are best utilized during the later downstream phases after requirements elicitation, such as during analysis and specification. There are a number of well known problems with using natural language for requirements specifications, but using a more formal notation so early on in the project runs the risk of making the requirements impossible to understand for the stakeholders (Durán Toro et al. 1999). In a survey of nine of the top requirements elicitation experts conducted by Hickey and Davis (Hickey & Davis 2003b), none of them mentioned using formal methods during elicitation. This is not surprising given that natural languages are in most cases the only ones understandable to end users.

The success of a system is heavily dependent on the quality of the requirements used for its development, and in turn on the process used to elicit them. The quality of the elicitation process can subsequently be expressed in terms of the correctness, completeness, consistency, and clarity of the resultant elicited requirements. Furthermore requirements should be concise, unambiguous, understandable, identifiable, readable, traceable, prioritized, organized, modifiable, verifiable, and ultimately feasible and useable (Davis, A. M. 1993). Other commonly used quality attributes for requirements include their relevance to the scope of the project, the extent to which they are feasible given the constraints of the project, and the ability to trace their source and rationale. It is also important that requirements are stated in such a way as they can be tested to determine their quality and if they have been fulfilled (Lauesen 2002). In reality some element of ambiguity, contradiction, and incompleteness is inevitable and accepted during requirements elicitation given that this represents only the first phase in the RE process. However the success of the requirements elicitation process can really only be measured by the quality of the requirements produced and the satisfaction of the participants involved in the process (El Emam & Madhavji 1996).

The question of when a requirements elicitation process should finish, and how much is enough requirements elicitation, is once again very dependent on the specific project. The desire to create a perfect set of requirements must be balanced with the realistic goal of producing a system within a reasonable timeframe. Therefore in reality its completion is often determined by time and cost constraints rather than achieving the required level of abstraction and requirements quality. Ultimately the how and when to end the process is really a judgment call by the analyst that, takes courage, and should be based on the contextual situation of the specific project, and should be agreed to by all the participants (Gause & Weinberg 1989). Al Davis in his work on 'Just Enough Requirements Management' (Davis, A. M. 2004) has addressed this specific issue directly and in detail.

2.3.9 Section Discussion

It is generally acknowledged that some form of process guidance is necessary for effective and efficient requirements elicitation. One study suggests that the non-use of a process methodology for requirements elicitation results in more iterations, and greater time and costs (Chatzoglou 1997). As we saw in Section 2.3.3, the current process models on offer for requirements elicitation consist of mainly generic lists of high-level activities. These models lack definitive guidelines and roadmaps for their structured and rigorous execution, and as a result, provide only very limited assistance to analysts. In fact Hickey and Davis from their extensive work on the subject of requirements elicitation point out that most process models focus on either a) a specific methodology or technique, or b) only model requirements elicitation in general terms (Hickey & Davis 2003c). They go on to say that the first class of models "possess a variety of weaknesses including the fact that each only describes a specific elicitation methodology or technique, each prescribes a specific series of steps with its own predefined technique, and each fails to model either the technique selection process or the situational characteristics that drive that decision process". In addition, for the second class, they identify that the models "possess different weaknesses such as most have underlying, but un-stated, assumptions, and none discuss the role of knowledge in performing requirements elicitation with respect to the current problem, solution, and project characteristics".

Consequently, we suggest that what is really needed is a structured and rigours process model for requirements elicitation that is both useful at a detailed level and useable for the majority of analysts, but is sufficiently flexible to allow for different situational project characteristics and not restricted to any specific method. As Chatzoglou states "since most of the existing methodologies are general purpose and do not utilize the characteristics of the specific domain, what is needed is a customization of the whole development process to suit particular projects needs and characteristics" (Chatzoglou 1997). Furthermore, this process model should lead to improvements in both effectiveness and efficiency. Although requirements elicitation is undoubtedly one of the most difficult software development processes to model (Hickey & Davis 2003c), a major advantage of developing and employing such a rigorous and structured process towards requirements elicitation is the ability to produce standardised and maintainable outputs (Macaulay, L. 1996). In addition, it is anticipated that a defined process will make it possible to estimate the effort, time, and cost required more accurately, something that is virtually impossible if an ad-hoc method is adopted.

In summary, the key points identified in this section are:

1. The process of requirements elicitation is highly dependent on the context

2. Current process models for requirement elicitation provide little actual guidance

3. Understanding the domain is an important first step in requirements elicitation

4. Requirements come from a variety of sources including human stakeholders

5. Method selection is contextual and effects requirements quality

6. A combination of methods are necessary for successful requirements elicitation

7. Detailed yet situational process models are required

2.4 Requirements Elicitation Techniques

2.4.1 Section Overview

As the name implies, a requirements elicitation technique is a technique used by the analyst to elicit requirements from stakeholders and other sources. More generally a 'technique' is a way of doing something or a practical method applied to some particular task (Farlex 2006). A technique is intended to provide guidance for both the analyst and stakeholders in eliciting requirements, in order to avoid the 'blank slate syndrome' common when people are asked to produce information (Moore & Shipman 2000). In reality there are literally hundreds of available techniques from a variety of sources that can be employed for requirements elicitation. An early survey by Goguen and Linde (Goguen, J. A. & Linde 1993) examined at a relatively high level only a small number of the more traditional techniques such as interviewing, observation, and task analysis. In a more recent survey on the theory and practice of requirements elicitation (Zowghi & Coulin 2005), several additional and more current approaches were examined including those based on goals, scenarios, viewpoints, and domain knowledge.

In this section we present just some of those techniques that are more widely used, in order to evaluate their relative strengths and weaknesses in eliciting requirements and addressing the current issues. Although this selection is by no means complete or exhaustive, we believe it is representative of the range of available techniques described in the literature and performed in practice today. Despite the difficulty, but for the purposes of presentation and completeness, we have also attempted to categorise these selected techniques. Not surprisingly there is any number of criteria by which different techniques may be classified and grouped together, and many of the selected techniques can easily be associated with more than one of the resultant groups. Therefore, for the sake of simplicity and understanding, we have based our categorisation on what we have considered to be the most obvious characteristic of each technique, which also has some commonality with two or more of the other selected techniques. We have also taken into consideration, and tried to remain

consistent where possible and when appropriate, the classes of requirements elicitation techniques distinguished by Nuseibeh and Easterbrook (Nuseibeh & Easterbrook 2000). This classification has been cited elsewhere in the literature (e.g. (Tuunanen & Rossi 2004)), and provides a suitably broad and logical coverage of the available requirements elicitation techniques for the purposes of evaluation and comparison.

2.4.2 Traditional Techniques

Traditional requirements elicitation techniques include a broad class of generic information gathering techniques (Nuseibeh & Easterbrook 2000). These techniques are those which have been used since the beginnings of software engineering for the purposes of determining the needs and wants of customers and users, even before requirements elicitation had been established as a separate area of interest within computer science. In fact most of these techniques have been adopted from other disciplines such as the psychology and sociology.

2.4.2.1 Interviews

Interviews (Agarwal & Tanniru 1990; Holtzblatt & Beyer 1995) are probably the most traditional and commonly used technique for requirements elicitation by analysts. Because interviews are essentially human based social activities, they are inherently informal and their effectiveness depends greatly on the quality of interaction between the participants. Interviews provide an efficient way to collect large amounts of data quickly from groups or individuals. The quality of the results from interviews, such as the usefulness of the information gathered, can vary significantly depending on the skill of the interviewer (Goguen, J. A. & Linde 1993). In basic terms there are fundamentally three types of interviews being unstructured, structured, and semi-structured, the latter generally representing a combination of the former two.

Unstructured interviews are conversational in nature where the interviewer enforces only limited control over the direction of discussions. Because they do not follow a

predetermined agenda or list of questions, there is the risk that some topics may be completely neglected. It is also a common problem with unstructured interviews to focus in too much detail on some areas, and not enough in others (McGraw & Harbison-Briggs 1989). Because of their dynamic nature, unstructured interviews require a significant amount of skill to be performed well. Furthermore, the large amount of resultant data can make them hard to analyse and compare. This type of interview is best applied for exploration when there is a limited understanding of the domain, or as a precursor to more focused and detailed structured interviews.

Structured interviews are conducted using a predetermined set of questions, either open-ended or closed, to gather specific information. The success of structured interviews depends on knowing what are the right questions to ask, when should they be asked, and who should answer them. Templates that provide guidance on structured interviews for requirements elicitation such as Volere (Robertson, S. & Robertson 1999) can be used to support this technique. Although structured interviews tend to limit the investigation of new ideas (i.e. the breadth, depth, flexibility and allowance for spontaneity is lower than unstructured interviews), they are generally considered to be rigorous and effective. These are more suitable for novice analysts as their evaluation is easier, and they require limited training and require less time (Kendall & Kendall 2002).

Although dependent on the purpose, in general terms interviews should start with basic and high-level topics, and then focus on the specifics of each relevant item. Because of the ability to elicit rich information, probe, and follow-up, interviews are generally considered to be good for discovering opinions, feelings, goals, attitudes and beliefs, particularly with respect to current tasks and issues. However the problem with interviews for requirements elicitation is that they can be costly and time consuming in terms of preparation, execution, and analysis, and typically it is necessary to interview multiple stakeholders several times in order to obtain and produce a set of quality requirements. This is mainly because each user that is interviewed will tend to focus on the system from their own individual perspective.

2.4.2.2 Questionnaires

Questionnaires (Foddy 1994) are mainly used during the early stages of requirements elicitation and may consist of open-ended and/or closed questions. For them to be effective, the terms, concepts, and boundaries of the domain must be well established and understood by the participants and questionnaire designer. Questions must be focused to avoid gathering large amounts of redundant and irrelevant information. They provide an efficient way to collect large amounts of information from multiple stakeholders quickly, and can easily be performed remotely (e.g. online). However they are limited in the depth of knowledge they are able to elicit, and in general provide little supporting contextual information. Most importantly, questionnaires lack interactivity, and consequently do not provide the opportunity to delve further on a new topic or expand on fresh ideas as interview do. In the same way they provide no mechanism for the participants to request clarification or correct misunderstandings. Furthermore, the design of questionnaires can be difficult as open-ended questions results can be hard to analyse, and the results from closed questions can be easily misinterpreted. Generally questionnaires are considered more useful to determine basic attitudes, beliefs, and characteristics, as informal checklists to ensure fundamental elements are addressed early on, and to establish the foundation for subsequent elicitation activities.

2.4.2.3 Task Analysis

Task analysis (Carlshamre & Karlsson 1996; Richardson, Ormerod & Shepherd 1998) employs a top-down approach where high-level tasks are decomposed into subtasks and eventually into detailed sequences until all actions and events are described. The primary objectives of this technique is to construct a hierarchy of the tasks performed by the users and the system, and determine the knowledge used or required to carry them out. Task analysis provides information on the interactions of both the user and the system with respect to the tasks as well as a contextual description of the activities that take place. In most cases considerable effort is required to perform thorough task analysis, and it is important to establish what level of detail is required and when

components of the tasks need to be explored further. Despite this task analysis is a useful technique to employ in order to investigate useability problems.

2.4.2.4 Domain Analysis

Examining related documentation and applications in the target domain of the system is a very useful technique for gathering early requirements and identifying reusable concepts and components. These types of investigations are particularly important when the project involves the replacement or enhancement of an existing legacy system, or when the analyst is not familiar with the organisation. Types of documentation that may be useful for eliciting requirements include design documents and instruction manuals for existing systems, and hardcopy forms and files used in the current business processes. This analysis can provide good background information about the organisation, the business, and its processes. However the available documentation may contain large amounts of irrelevant information, or worse, may be inconsistent with the real operations. Application studies often also include looking at both upstream and downstream systems, as well as the analysis of competitive or like products and solutions. This may extend to the reverse engineering of existing systems with respect to what is being used, what is not, what is missing, what is good, and what is bad. In most cases these studies involve other elicitation techniques such as observing the exiting system in use and interviewing the current users.

Domain knowledge in the form of detailed descriptions and examples on the other hand can also play an important part in the process of requirements elicitation. Approaches based on this type of information are often used in conjunction with, and as the input for, other elicitation techniques. For example analysts use previous experience in similar domains as a discussion template for facilitating group work and conducting inter-views. Analogies and abstractions of existing problem domains can be used as baselines to acquire specific and detailed information, identify and describe possible solution systems, and assist in creating a common understanding between the analyst and stakeholders. These approaches also provide the opportunity to reuse specifications and validate new requirements against other domain instances (Sutcliffe, A. & Maiden 1998). Problem Frames (Jackson 2000) in particular provide

a method for detailed problems examination in order to identify patterns that could provide links to potential solutions.

2.4.2.5 Introspection

The technique of introspection (Goguen, J. A. & Linde 1993) requires the analyst to develop requirements based on what he or she believes the users and other stakeholders want and need from the system. Despite being employed by most analysts to some extent, this technique is mainly used only as a starting point for other requirements elicitation efforts. Introspection is only really effective when the analyst is not only very familiar with the domain and goals of the system, but also expert in the business processes performed by the users. In cases where the analyst is forced to use this technique more, for example when the users have little or no previous experience with software systems in their work environment, a type of facilitation introspection should take place via other elicitation techniques such as interviews and protocol analysis. Given that when using this technique the analyst is required to project what the stakeholders think, introspection has the potential to be highly inaccurate (Goguen, J. A. & Linde 1993).

2.4.3 Cognitive Techniques

Cognitive techniques include a series of techniques originally developed for knowledge acquisition (Nuseibeh & Easterbrook 2000). These techniques aim to elicit requirements by representing and structuring the knowledge of stakeholders in terms of how they see both the problem and solutions domains.

2.4.3.1 Card Sorting

Card sorting requires the stakeholders to sort a series of cards containing the names of domain entities into groups according to their own understanding. Furthermore the stakeholder is required to explain the rationale for the way in which the cards are sorted. It is important for effective card sorting that all entities are included in the process. This is possible only if the domain is sufficiently understood by both the

analyst and the participants. If the domain is not well established then group work can be used to identify these entities. This technique is often used more for the categorization and clarification of requirements rather than elicitation. Class Responsibility Collaboration (CRC) cards (Beck & Cunningham 1989) are a derivative of card sorting that is used also to determine program classes in software code. In this technique cards are used to assign responsibilities to users and components of the system. Because entities represent such a high level of system abstraction, the information obtained from this technique is limited in its detail.

2.4.3.2 Laddering

When using laddering (Hinkle 1965), stakeholders are asked a series of short predefined prompting questions known as ‘probes’, about one or more concepts related to the target system, and are then required to arrange the resultant answers into an organized structure according to their own understanding and preferences. This knowledge, which is often displayed using tree diagrams of interlocking ladders, is then reviewed and modified dynamically as more information is added. Much like card sorting, laddering is mainly used as a way to clarify requirements and categorize domain entities. A primary assumption when employing laddering is that the knowledge to be elicited can actually be arranged in a hierarchical fashion. For this technique to be effective, the stakeholders must be able to express their understanding of the domain and then arrange it in a logical way.

2.4.3.3 Repertory Grids

Repertory grids (Kelly 1955) involve asking stakeholders to develop attributes and assign values to a set of domain entities. As a result the system is modelled in the form of a matrix by categorizing the elements of the system, detailing the instances of those categories, and assigning variables with corresponding values to each one. The aim is to identify and represent the similarities and differences between the different domain entities. These represent a level of abstraction unfamiliar to most users. As a result this technique is typically used when eliciting requirements from domain experts. Although more detailed than card sorting, and to a lesser degree laddering,

repertory grids are somewhat limited in their ability to express specific characteristics of complex requirements.

2.4.4 Group Techniques

Group elicitation techniques aim to foster stakeholder agreement and buy-in, while exploiting team dynamics (Nuseibeh & Easterbrook 2000). They typically require three or more stakeholders working together in order to generate ideas and specifications for the target system.

2.4.4.1 Brainstorming

Brainstorming (Osborn 1979) is a process where participants from different stakeholder groups engage in informal discussion to rapidly generate as many ideas as possible without focusing on any one in particular, where quantity is paramount and not quality. This is typically followed by a consolidation stage where the number of ideas is narrowed down by removing those ideas that the group immediately identifies or recognises as inappropriate or unsuitable, and then the remaining ideas are examined and evaluated, refining and combining them until the group is satisfied with the results. It is important when conducting this type of group work to avoid exploring, critiquing or analysing ideas in detail. All persons present should actively participate with equal worth, be creative, and all the ideas generated should be recorded, no matter how unrealistic they may seem. It is not usually the intended purpose of brainstorming sessions to resolve major issues or make key decisions. This technique is often used to develop the preliminary mission statement for the project and target system. One of the advantages in using brainstorming is that it promotes freethinking and expression, and allows the discovery of new, creative, and innovative solutions to existing problems.

2.4.4.2 Requirements Workshops

Requirements workshop (Gottesdiener 2002) is a generic term given to a number of different types of group meetings where the emphasis is on developing and

discovering requirements for a software system. There are many different forms of requirements workshops depending on their purpose and participants. Workshops are a well established, very common and often default technique for requirements elicitation. Groups are particularly effective because they involve and commit the stakeholders directly and promote cooperation. These types of sessions can be difficult to organize due to the number of different stakeholders that may be involved in the project. Managing these sessions effectively normally requires a highly trained facilitator with both expertise and experience to ensure that individual personalities do not dominate the discussions. This facilitation is very important, as with all social interactions of people from different and varied backgrounds, there needs to be some level of overriding control in order to prevent chaos and anarchy. Gottesdiener (Gottesdiener 2001) recommends the use of 'collaborative patterns' such as 'Divide, Conquer, Correct, Collect', as well as walkthroughs and checklists can be used to further enhance requirements workshops.

Key factors in the success of group work are the makeup of participants and the cohesion within the group. Participants should be motivated and cooperative, have the right mix of skills and knowledge, and should all share a common goal (Gottesdiener 2001). Stakeholders must feel comfortable and confident in speaking openly and honestly, and it is for this reason that group work is less effective in highly political situations. In all cases, many social factors need to be considered and managed when using workshops to elicit requirements, and it is important to be careful of bias, dominance, and submission of participants. Workshops are often performed using support materials such as documents, diagrams, and prototypes to promote discussion and feedback. This technique encourages stakeholders to resolve conflicts and develop solutions themselves, rather than relying on the analyst to drive the process. In addition, groups have the ability to allow more natural interactions between people such as free flowing conversation, as oppose to the question and answer format of interviews and questionnaires. One of the other obvious advantages of using group workshops is the ability to integrate other elicitation techniques into them, and then to incorporate their combined usage into a defined requirements process (Maiden, N. et al. 2004).

2.4.4.3 Focus Groups

Another variation of requirements group work derived and often used in market analysis and subsequently market-driven systems development is Focus Groups (Krueger 1994). This technique is a kind of group interview, with a session lasting not last more than two hours and involving around a dozen people at the most. Typically focus groups are used to discuss a specific topic or address a particular problem. In software development, focus groups often use stimulus material such as questionnaires, prototypes, or storyboards to provoke and encourage dialog among the participants. Because the facilitator usually takes a passive role during the session, and the level of structure required is flexible, focus groups allow more natural conversation than more defined group work approaches. Providing that the participants are expert in the area of discussion, focus groups can also enable the elicitation of important, useful, and accurate opinions and perceptions about the target system. Although relatively practical and economical to conduct, this technique is not well suited to finding solutions to complex problems or eliciting hard requirements.

2.4.4.4 Creativity Sessions

It is generally accepted that creativity and imagination are essential components in successful problem solving, and enable the generation of unexpected solutions to difficult problems (Mich, Anesi & Berry 2005). As a result the concepts of inventing and originating requirements have been used to highlight the role of innovative thinking and expression in elicitation, and to emphasize what really goes on during process (Maiden, N., Gizikis & Robertson 2004; Robertson, J. 2002). The aim of creativity session within the context of requirements elicitation is to discover missing and develop new requirements that would not be elicited using more traditional and conservative techniques. Providing the group with stimulation during these sessions is essential, and there are literally hundreds of games and methods specifically designed to encourage participants to flex their creative muscles. Most of these come from the social sciences, and include things like ‘Mindmapping’ (Wycoff 1991) and ‘Six Thinking Hats’ (de Bono 1999) to name but two of those most widely known. In order for creativity sessions to be effective, groups must be motivated, active, and open-

minded. Because typically very little structure is used when applying this technique, as is generally necessary for creative thought and discussion, these sessions can be hard to manage, and the results are often unpredictable.

2.4.4.5 Nominal Group Technique

The Nominal Group Technique (Delbecq, Van de Ven & Gustafson 1975) represents a more structured form of brainstorming or brain writing. The process generally begins with the participants generating ideas in writing but anonymously based on a problem stated by the facilitator. Each person then reads out one idea in turn in a round-robin fashion recorded by the facilitator, until all ideas have been presented. The group then works through each idea in sequence to clarify its details, ask questions, and offer comments, thereby creating a shared understanding for each idea. The ideas are then voted upon anonymously as to their importance and/or relevance to the problem. The steps of discussion and voting may continue to take place as necessary several times until a general decision is achieved. This technique is particularly useful in politically or socially sensitive situations as it attempts to balance the influence of all the individual participants, thereby reducing somewhat the negative effects of group dynamics. Furthermore it encourages the participants to seek a group solution as a process of problem solving rather than by negotiation. Although effective for consensus building, the Nominal Group Technique tends to be limited to a single purpose and single topic meeting, and requires significant preparation and strong facilitation. This can be used as an alternative to the Delphi technique (Helmer-Hirschberg 1967) and focus groups, and has been integrated within other types requirements elicitation workshops include JAD (Duggan & Thachenkary 2004).

2.4.5 Contextual Techniques

Contextual techniques have been utilized for requirements elicitation as an alternative to both traditional and cognitive techniques (Nuseibeh & Easterbrook 2000). These techniques focus on gathering the requirements directly from the context in which the target system will eventually exist, that is the specific environment in the real world.

2.4.5.1 Ethnography

Ethnography (Ball & Ormerod 2000; Sommerville 2001) being the study of people in their natural setting is a form of social analysis and involves the analyst actively or passively participating in the normal activities of the users over an extended period of time whilst collecting information on the operations being performed. This technique in its various forms is especially useful when addressing contextual factors such as useability, and when investigating collaborative work settings where the understanding of interactions between different users with the system is paramount. Ethnography is particularly effective when the need for a new system is a result of existing problems with processes and procedures, and in identifying social patterns and complex relationships between human stakeholders. Some, such as Gougen and Linde (Goguen, J. A. & Linde 1993), have suggested that ethnography should be used throughout the requirements elicitation process to provide the contextual basis for the results of other elicitation techniques.

2.4.5.2 Observation

Observation (Wixon & Ramey 1996) is one of the more widely used contextual techniques, and as the name suggests, the analyst simply observes the actual execution of existing processes by the users without direct interference. This technique is often used in conjunction with others such as interviews and task analysis. As a general rule, observation is expensive to perform, and requires significant skill and effort on the part of the analyst to interpret and understand the actions being performed. The effectiveness of observation can vary as users have a tendency to adjust the way they perform tasks when knowingly being watched. Furthermore, observational studies may need to be carried out over a long period of time in order for the analyst to be able fully understand what is actually taking place. Despite this, observation is considered to be a good technique for generalizing knowledge about current operations and associated issues, however this also create a bias towards what is presently being done.

2.4.5.3 Protocol Analysis

Protocol analysis (Goguen, J. A. & Linde 1993; Nielsen, Clemmensen & Yssing 2002) is where participants perform an activity or task whilst talking it through aloud, describing the actions being conducted and the thought process behind them. This technique can provide the analyst with specific information on and rationale for the processes the target system must support (McGraw & Harbison-Briggs 1989). In most cases however talking through an operation is not the normal way of performing the task, and as a result may not necessarily represent the true process completely or correctly. Likewise minor steps performed frequently and repetitively are often taken for granted by the users, and may not be explained and subsequently recorded as part of the process.

2.4.5.4 Apprenticing

Apprenticing (Beyer, H. R. & Holtzblatt 1995; Robertson, S. & Robertson 1999) involves the analyst actually learning and performing the current tasks under the instruction and supervision of an experienced user. In this technique the analyst is taught the operations and business processes by observing, asking questions, and physically doing, rather than being informed of them, as is the case with protocol analysis. Similar but more involved than Role Playing (Leffingwell & Widrig 2000), where the analyst simply takes the place of the user and performs the related work tasks, apprenticing is very useful when the analyst is inexperience with the domain, and when the users have difficulty in explaining their actions. The technique of apprenticing can be taken one step further whereby the analyst becomes immersed and actively involved in the real life activities of the business.

2.4.5.5 Prototyping

Providing stakeholders with prototypes of the system to support the investigation of possible solutions is an effective way to gather detailed information and relevant feedback (Sommerville 2001). It is common and advantageous to use prototypes in conjunction with other elicitation techniques such as interviews and group work to

encourage discussion and debate. Prototypes are typically developed using preliminary requirements or existing examples of similar systems. This technique is particularly useful when developing human-computer interfaces, or where the stakeholders are unfamiliar with the available solutions, and there is a great deal of uncertainty about the requirements (Nuseibeh & Easterbrook 2000). There are a number of different methods for prototyping systems such as storyboards, executable, throwaway and evolutionary, with varying levels of effort required. In many cases prototypes are expensive to produce in terms of time and cost. However, an advantage of using prototypes is that they encourage stakeholders, and more specifically the users, to play an active role in developing the requirements, as it is easier to discuss an actual tangible system. One of the potential hazards when using prototypes for requirements elicitation is that users may become attached to them, and therefore become resistant to alternative solutions from then on. Despite this the technique is extremely helpful when developing new systems for entirely new applications.

2.4.6 Section Discussion

Through our review we have seen that every technique has some limitations (Goguen, J. A. & Linde 1993), and even for the most common and natural one of interviews, there is insufficient available support for navigating the specifics of a requirements elicitation process, and no guidance for specifying the requirements (Kato et al. 2001). Goguen and Linde (Goguen, J. A. & Linde 1993) in their seminal work on requirements elicitation techniques present a very good and detailed description of the social problems and limitations of many of the requirements elicitation techniques we have reviewed. In it they are particularly scathing of protocol analysis, but very complimentary of discourse analysis in determining the ‘real’ social order that makes up part of the context of the target system. Unfortunately this type of ethnomethodological technique requires significant expertise and substantial investment in both time and effort. As stated by Moore and Shipman (Moore & Shipman 2000) “the problem is that requirements gathering methods tend to fall into two categories: those which produce rich results but are expensive (in time and money) and those that are less expensive but also less informative”. An important point is that although analysts may be familiar with several requirements elicitation

techniques, no one technique in isolation is able to capture all the requirements completely (Maiden, N. A. M. & Rugg 1996). Therefore more than one technique is needed to elicit all the actual requirements for complex software-based systems.

We have shown that there exists a wealth of requirements elicitation techniques available, and because of this wide range of techniques it is possible to use alternative techniques in many situations, which enable greater flexibility of the process and more choice for the analysts and stakeholders. It can also been seen that many of these techniques do not originate from the traditional areas of Software Engineering or Computer Science research. Techniques for requirements elicitation are derived mostly from the social sciences, organisational theory, group dynamics, knowledge engineering, and very often from practical experience. As a result, some techniques are good at eliciting domain knowledge, or user knowledge, or current and future situations. Most requirements elicitation techniques are informal and involve human-to-human communication and interaction. Of all the techniques, group work is particularly effective as it would appear that groups are able to deal with complex tasks such as requirements elicitation better than individuals because they have a wider range of skills and abilities to draw from. Group work techniques are also beneficial because they involve the users, commit the customers, and promote discussion, collaboration, idea generation, solution finding, and decision making. Another advantage of Requirements Workshops is that they enable the incorporation of other requirements elicitation techniques. Furthermore group techniques are naturally very important to the requirements elicitation process because software development is inherently a group effort (Palmer & Fields 1992).

However there is some debate over the relative performance of facilitated workshops versus one-on-one interviews. In a recent study (Schalken, Brinkkemper & van Vlieet 2004) it was found that one-on-one interviews as prescribed by the Method/1 method (Arthur Andersen 1988), were found to be more efficient (i.e. greater productivity) for small projects, whereas facilitated workshops in accordance with the DSDM method (Stapleton 2002) were more efficient for larger projects. It is important to note that almost all the current techniques, including requirements workshops, brainstorming, and interviews, do not address requirements expression (Durán Toro et al. 1999), i.e. how to represent and present the collected information.

In summary, the key points identified in this section are:

1. Traditional techniques can be expensive and superficial

2. Cognitive techniques are for experts and provide only basic information

3. Group techniques are flexible and effective but require appropriate facilitation

4. Contextual techniques are expensive and require significant expertise

5. Few, if any, of the techniques offer detailed process support specifically for requirements elicitation

2.5 Requirements Elicitation Approaches

2.5.1 Section Overview

An ‘approach’ can be defined as an arrangement of ideas and/or actions intended to deal with a problem or situation (Farlex 2006). Unlike the techniques reviewed in the previous section, requirements elicitation approaches tend to be more specific to the actual task of eliciting requirements, or at least developing a specification for a software system. Once again there are literally dozens of approaches that can, and have, been used for requirements elicitation. We have therefore taken a representative sample of these approaches in order to present the major types of approach often used in the both research and practice.

The classification scheme used for the reviewed approaches is once more just for explanation purposes, and is of our own making in the absence of an accepted structure. As with the techniques reviewed in the previous section, some of the selected approaches may be classified into one or more of the classes identified. We have tried to identify commonalities between the available approaches reviewed, and utilize that as the basis for our categorisation. The purpose of this review, and our categorisation of requirements elicitation approaches, is therefore to evaluate and compare their relative advantages and disadvantages, in order to determine the key characteristics of successful approaches, and identify those areas of requirements elicitation that are in need of further attention.

2.5.2 Modelling Approaches

Model-driven approaches provide a specific model of the type of information to be gathered in order to drive the process (Nuseibeh & Easterbrook 2000). Models can be of different types including descriptive, graphical, and mathematical, and several approaches based on each of these types has been proposed as a way of performing requirements elicitation. Model driven approaches provide ways of representing the existing or future processes and systems using analytical techniques with the intention

of investigating their characteristics and limits. It has already been argued that many forms of modelling may also be considered as requirement elicitation techniques, especially those with easy to understand diagrammatic notations (Subramaniam 1999). It is certainly true that these are used in practice for requirements elicitation, and are often categorised as approaches to requirements elicitation. In fact historically modelling was the main elicitation technique, and heavily relied upon for this activity (Hickey & Davis 2003b).

2.5.2.1 Goals

Goal based approaches for requirements elicitation have become increasingly popular in both research and practice. The fundamental premise of goal modelling and goal-based approaches is that high-level goals that represent objectives for the system are decomposed (e.g. usually using AND and OR relationships) and elaborated (e.g. with 'Why' and 'How' questioning) into sub goals and then further refined and incrementally expanded in such a way that individual requirements are elicited. More specifically high-level goals are typically identified through discussion, with respect to what needs to be improved, achieved, avoided, or reduced. These are then organised by assigning a type and priority to each goal, and refined by describing the rationale for each one. The goals are then elaborated upon through their combination (composition) and separation (decomposition), and finally each goal is operationalised by continuing elaboration until the discussion starts to generate possible ways of achieving and satisfying these goals.

The result of this process is significantly more complicated and complete than the traditional methods of representing system goals using tree structure diagrams. These approaches are able to represent detailed relationships between domain entities, requirements, and the objectives of the system. In recent times significant effort has been devoted to developing these types of approaches for requirements elicitation such as the F^3 project (Bubenko & Wangler 1993), the KAOS meta model (Dardenne, van Lamsweerde & Fickas 1993) and the i* framework (Yu 1997). In practice these approaches have been particularly useful in situations where only the high-level needs for the system are well known, and there exists a general lack of understanding about

the specific details of the problems to be solved and their possible solutions. However one of the risks when using goal-based approaches is that errors in the high-level goals of the system made early on can have a major and detrimental follow on effect, and that changing goals are difficult to manage.

2.5.2.2 Scenarios

Scenarios are widely used in requirements elicitation and as the name suggests are narrative and specific descriptions of current and future processes, including actions and interactions between the users and the system (Alexander, I. F. & Maiden 2004). Scenarios typically include a description of the system state at the beginning of the process, a sequential flow of events given certain conditions, and a description of the system state at the end of the process. Scenarios do not typically consider the internal structure of the system, and require an incremental and interactive approach to their development. Naturally it is important when using scenarios to collect all the potential exceptions for each step, which in many cases requires significant effort. A substantial amount of work from both the research and practice communities has been dedicated to developing structured and rigorous approaches to requirements elicitation using scenarios including CREWS (CREWS 1999), The Inquiry Cycle (Potts, Takahashi & Anton 1994), SBRE (Kaufman, Thebaut & Interrante 1989), and Scenario Plus (Scenario Plus 2007). Scenarios are additionally very useful for understanding and validating requirements, as well as test case development. The use of scenarios in conjunction with goals to elicit requirements has also attracted considerable attention (Haumer, Pohl & Weidenhaupt 1998; Potts, Takahashi & Anton 1994; Rolland, Souveyet & Ben Achour 1998). One of the real advantages in using scenarios is that they are very flexible in being able to be used informally or with greater degrees of structure.

2.5.2.3 Use Cases

The popularity and acceptance of Use Cases for eliciting requirements has increased significantly in recent years thanks in part to their ‘integration’ with the Unified Modelling Language (UML) (OMG 2004). Use Cases are essentially a higher level of

abstraction from scenarios that describe the functional behaviour of the system (Alexander, I. F. & Maiden 2004), and as a result are more appropriate for the early stages of requirements elicitation. The diagrammatic and tabular representations of Use Cases (Cockburn 2001) make them particularly easy to understand and flexible enough to accommodate context specific information. Use cases can be reused later in the development process to determine components and classes during system design, and when creating test cases. This technique is especially effective in projects where there is a high level of uncertainty or when the analyst is not an expert in that particular domain. However, Use Cases have several shortcomings (Berenbach 2004), including the fact that they are not a precise specification, and do not capture domain knowledge very well. Storyboarding (Leffingwell & Widrig 2000) is another approach like Use Cases that identifies the players, explains what happens to them, and describes how it happens. However storyboards tend to be less structured, more graphical, and less specific with respect to the details of the user and system interactions.

2.5.2.4 Viewpoints

Viewpoint approaches aim to model the domain from different perspectives in order to develop a complete and consistent description of the target system. For example a system can be described in terms of its operation, implementation and interfaces. In the same way systems can be modelled from the standpoints of different users or from the position of related systems. Depending on the specific approach, viewpoints can be defined as data sources and sinks (CORE (Mullery 1979)), a type of system model (ViewPoints (Nuseibeh, Kramer & Finkelstein 1994)), or a receiver of services (VORD (Kotonya & Sommerville 1996)). These types of approaches are particularly effective for projects where the system entities have detailed and complicated relationships with each other. Viewpoints are also useful as a way of supporting the organisation and prioritization of requirements. One common criticism of viewpoint approaches is that they do not enable non-functional requirements to be represented easily, and are expensive to use in terms of the effort required. Some viewpoint approaches (Nuseibeh, Finkelstein & Kramer 1996; Sommerville, Sawyer & Viller 1998) provide a flexible multi-perspective model for systems, using different

viewpoints to elicit and arrange requirements from a number of sources. Using these approaches analysts and stakeholders are able to organize the process and derive detailed requirements for a complete system from multiple project specific viewpoints.

2.5.2.5 Dialogs

A number of approaches have been proposed based on modelling the dialog structure between analysts and users in order to navigate the process of eliciting system requirements including (Lecoeuche, Mellish & Robertson 1998), (Kato et al. 2001), and to a lesser extent (Leite & Gilvaz 1996). In particular Kato et al. analysed the interview processes of experts and explored a computational model for simulating them. In the resultant approach, a mixture of models is used including a blackboard and state transition model for selecting the questions to ask, and a thesaurus as a way of providing topics for conversation in the interview process. The state transition model specifies 1) which type of questions should be asked in what order, 2) what type of information can be received as answers, and 3) in which slots on the template the answers should be written. This approach and others like it therefore sacrifice some of the benefits of natural language conversation during the requirements elicitation process, for the sake of more structured and rigorous guidance. However, for the most part these approaches have been limited in scope and application, and have not been evaluated in real-world projects, or even fully implemented towards solving practical problems.

2.5.3 Combinational Approaches

Because of the relative strengths and weaknesses of different requirements elicitation techniques, and the type of information they provide, the reality is that in almost all projects a combination of several different techniques will be necessary to achieve a successful outcome and the best possible results (Maiden, N. A. M. & Rugg 1996). This is supported by the fact that many requirements elicitation techniques can be used in conjunction with each other to address particular problems (Goguen, J. A. & Linde 1993). As a result, a number of approaches have been developed that combine

complementary requirements elicitation techniques, as detailed below. This type of approach can prevent the analyst from having to reshape the problem to fit a specific technique, which can lead to incorrect requirements and a disparity between the real problem and the proposed goals and requirements.

2.5.3.1 Zooming

Gougen and Linde (Goguen, J. A. & Linde 1993) propose that the more expensive requirements elicitation techniques in terms of time and effort should only be used selectively to examine in greater detail those needs deemed especially important, referring to this approach as ‘zooming’. One suggestion they offer along these lines and for the combination of techniques recommends that the process should begin with a general ethnographic study to discovery fundamental aspects of existing social patterns and behaviour, followed by structured interviews to gain deeper insight into the needs of the key stakeholders and the priorities of the core requirements. This approach attempts to tackle the individual weakness of otherwise effective techniques, and at the same time take advantage of their combined strengths, without incurring significant and unnecessary increases in both time and effort. The use of techniques such as discourse and interaction analysis is also suggested to investigate more deeply issues of critical value. As this approach provides only very general and high level advice on the combination of requirements elicitation techniques, substantial experience and expertise from the analyst would be required to plan and implement it successfully.

2.5.3.2 The Inquiry Cycle

Potts, Takahashi, and Anton (Potts, Takahashi & Anton 1994) have based their approach called the Inquiry Cycle on the close integration of goals and scenarios in order to perform requirements elicitation. The approach is based on a cyclical model consisting of three iteratively repeated steps being expression, discussion, and evolution. Scenarios are analysed to elicit requirements, which are then checked against identified system goals to ensure relevance and accuracy. This approach uses a strong foundation of continuing questioning and answering between the analyst and

the stakeholder throughout the process, therefore adding an informal interview to the combination of requirements elicitation techniques explicitly stated. Although not a rigid or formal process, the use of scenarios and goals provides the stakeholders with an understanding of the behaviour of the system within the context of solving the actual problems that need to be addressed. Another example of using this same combination can be found in (Rolland, Souveyet & Ben Achour 1998), which uses scenarios to guide goal modelling.

2.5.3.3 SCRAM

In (Sutcliffe, A. 1997) and (Sutcliffe, A. & Ryan 1998), the SCRAM approach is presented which proposes the combination and use of scenarios and with early prototypes called concept demonstrators, together with design rationale in order to elicit a complete, clear, and correct picture of the target system. These are integrated by a walkthrough method in order to guide the process of requirements elicitation through continuous questioning of the users, and then in return commenting and critiquing what is being presented. This is an iterative process as both the scenarios and prototypes are revised based on the feedback and design rationale provided. Because scenarios and prototypes present actual representations of the real world, the analyst is able to avoid the need to use technical language, thereby reducing the communication gap with the stakeholders. The use of scenarios as examples of real work processes is an attempt to reduce the negative effects of using prototypes, however stakeholder bias for the solutions implemented in the prototypes is somewhat unavoidable, as is the restriction of possible solutions using prototypes creates. The initial requirements for the prototypes are still elicited using conventional elicitation techniques, but many more functional and useability requirements are discovered as a result of this approach. Although this combination was found to increase stakeholder participation and understanding, it was acknowledged that the analyst style and skill could have a considerable impact on the quality of requirements elicited. In (Mannio & Nikula 2001) another example of the scenario and prototype combination based method is presented specifically for software requirements elicitation.

2.5.3.4 Critical Success Chains (CSC)

Another combinational approach originating from the Information Social Sciences and introduced in recent times for the elicitation of requirements is that of Critical Success Chains (CSC) (Peffers, Gengler & Tuunanen 2003). This is a top-down approach that combines two existing theories, namely Critical Success Factors and Personal Construct Theory, and consists of four basic phases being 1) Pre-study Preparation, 2) Data Collection, 3) Analysis, and 4) Ideation Workshops. This approach is similar to that of goal modelling except it is performed on a more individual and personal level, and aims to elicit rich information rather than just basic descriptions, however the use of graphical models to illustrate what is important about the target system is common. Recently this approach has been found to be useful in addressing more market-oriented requirements elicitation activities (Tuunanen & Rossi 2004). This included the determining of wants from potential customers rather than from end users, as is the case in the traditional process of requirements elicitation.

2.5.3.5 Best Practice Guides

A number of approaches to requirements elicitation in particular, and Software Engineering in general, have been based on the combination of multiple techniques that represent recommended best practices that can be selectively and dynamically applied to different project types and situations. Both IEEE's Software Engineering Body of Knowledge (SWEBOK) (IEEE 2004) and the REDEST project and case book (REDEST 2003) provide examples of this type of approach based on a collection of best practices found to have been successful or at least useful in industry. Similarly the OPEN Process Framework (OPF) (OPEN Process Framework 2007) provides a toolkit for requirements elicitation and other software development activities in the form of a repository of process components, guidelines, and checklists which can be assembled and combined as the analyst wants or the situation needs. Once again significant experience and expertise on the part of the participating analysis is assumed, and little if any guidelines for addressing contextual issues and managing situational factors.

2.5.4 Collaborative Approaches

Collaborative approaches as the name would imply are a collection of methods whereby a group of stakeholders work together cooperatively to develop the requirements for a target system. Unlike the group work techniques reviewed in the previous section, collaborative approaches tend to be more structured, with specific roles and tasks, however much of what has been said of Requirements Workshops in the previous section also applies. Collaboration allows stakeholders to be given some degree of ownership and control in the project in order to maintain their interest and provide incentives for participation and success.

2.5.4.1 Joint Application Development (JAD)

Joint Application Development (JAD) (August 1991; Wood, J. & Silver 1995) involves all the available stakeholders investigating through general discussion both the problems to be solved, and the available solutions to those problems. With all parties represented, decisions can be made rapidly and issues resolved quickly through the support for joint problem solving. Originally developed at IBM, JAD is a very important part of the Rapid Application Development (RAD) software engineering methodology. A major difference between JAD and brainstorming is that typically the main goals of the system have already been established before the stakeholders participate. Also JAD sessions are typically well structured with defined steps, actions, and roles for participants (including a specialist facilitator). The focus of this type of meeting tends to often be on the needs and desires of the business and users rather than technical issues. Potential benefits of JAD include the potential for time-savings, improved ownership, and creative development (Kendall & Kendall 2002). However they require a considerable commitment from all participants and the results can be unpredictable. Like all other forms of group work, the success of JAD as a collaborative approach to requirements elicitation depends on the cooperation and skills of the analyst and stakeholders.

2.5.4.2 PIECES

The PIECES (Performance, Information and data, Economy, Control, Efficiency, and Services) framework (Raghavan, Zelesnik & Ford 1994) was developed by the Software Engineering Institute (SEI) and is another collaborative approach to requirements elicitation. More structured than brainstorming, PIECES is similar to semi-structured interview but for groups of stakeholders, where a set of issues for each of the six categories is used to provoke the joint investigation and cooperative discussion. This limited guidance, which is more like a high level checklist to ensure the relevant subjects are covered, is intended to provide some degree of direction for the participants without restricting the scope or path of the conversations. Because this approach relies on the basic systems knowledge of the stakeholders, PIECES is best used when there is an existing system for reference, such as in the case or replacement or maintenance projects.

2.5.4.3 Creative Problem Solving (CPS)

Creative Problem Solving (CPS), or the Osborn-Parnes Creative Problem Solving Model (Parnes 1967) as it is also known, is an well-established creativity approach whose framework consists of six linear steps being 1) Objective Finding, 2) Fact Finding, 3) Problem Finding, 4) Idea Finding, 5) Solution Finding, and 6) Acceptance Finding. In addition to Brainstorming, the approach proposes that each step consists of a divergent thinking phase and a convergent thinking phase in order to elicitation information and solve problems. The CPS method does not specify any particular technique to accomplish each step in the method. Although CPS can be applied by individuals, its implementation within a collaborative group environment is the most beneficial, as problems can generally be solved more effectively as a collective task. In (Maiden, N., Gizikis & Robertson 2004) the CPS was used effectively to provide the framework for ordering workshop activities specifically for requirements elicitation, and particularly in seeking shared objectives and acceptances. The same was found in (Maiden, N. et al. 2004) where the CPS model worked well for requirements elicitation, providing finer-grain process guidance with which to structure each workshop session.

2.5.4.4 Cooperative Requirements Capture (CRC)

Cooperative Requirements Capture (CRC) (Macaulay, L. A. 1993) is a collaborative workshop based approach to requirements elicitation for group sessions similar to JAD, where there is a defined set of activities and led by a trained facilitator. CRC workshops are typically cross-functional, involving representation from all the different types of stakeholders from various areas of the business not just customers and developers. The CRC process includes activities such as Problem Identification, Team Selection, Exploring Users and User Environments, and Identification and Validation of the Scope. The focus of CRC is on the relationship between the users and their environments, and how they perform their tasks within those environments. Typically designers or developers will present ideas for the new system to the other stakeholders in order to create a share vision. The strength of CRC is that by examining the problem within the context of the business and normal operations, a shared and understandable picture of this problem domain is achieved. CRC is particularly effective for developing generic products, however the selection of the stakeholder representatives like most collaborative approaches is important to its success.

2.5.4.5 DSDM

DSDM (Dynamic Systems Development Method) (DSDM Consortium 2006) is a framework for business centred development, produced by an international non-profit consortium of industry and academic partners. Facilitated workshops consisting of participants with defined roles in the project are a core technique in DSDM, however many others are integrated including timeboxing (Martin, J. 1991) and prototyping. The entire approach is based on nine underlying principles being 1) Active user involvement is imperative, 2) The team must be empowered to make decisions, 3) The focus is on frequent delivery of products, 4) Fitness for business purpose is the essential criterion for acceptance of deliverables, 5) Iterative and incremental development is necessary to converge on an accurate business solution, 6) All changes during development are reversible, 7) Requirements are base-lined at a high

level, 8) Testing is integrated throughout the lifecycle, and 9) Collaboration and cooperation between all stakeholders is essential. In DSDM the project timeline and resources are fixed as early and as far as possible, and it is the requirements that will be satisfied which are changed dynamically. In this way it is claimed that DSDM improves time to market and the likelihood of projects being delivered on budget. Because requirements elicitation through facilitated workshop is very much client focused, the strengths of this approach are that user ownership and satisfaction is increased, and the risk of delivering the wrong solution is significantly reduced.

2.5.5 Methodological Approaches

A number of Software Engineering, and more specifically system analysis and design, methodologies exist that address requirements elicitation, typically through the integration of techniques to a standard set of tasks with support guidelines. Although these methodologies do not exclusively address requirements elicitation, or in many cases do not address it as a separate phase, they do however provide various mechanisms that can and have been used for it in research and practice.

2.5.5.1 Structured Analysis and Design (SAD)

Structured Analysis and Design (SAD) (DeMarco & Plauger 1979; Yourdon 1989) has been around since the mid 1970's and has been widely written about, promoted, and used. The approach is largely function oriented. It comprises of a collection of techniques such as Data Flow Diagrams (DFD) which detail the functional decomposition with the emphasis on the data in and out of the system and related components, and Entity Relationship Diagrams (ERD) that facilitate the representation of system entities, their attributes, and their relationships to each other. These are of particular interest for requirements elicitation as DFDs can be used to describe the high-level functional operations of the systems, and ERDs can be used to conceptually model the system. Other SAD techniques used during requirements elicitation include State Charts, Data Dictionaries, Event Lists, Decision Tables, and Decision trees, all of which can be used to provide supporting information for the system requirements.

In actuality there are a number of different SAD approaches (Yeates & Wakefield 2004) including generic approaches, which are relatively consistent with traditional system analysis and design approaches. Typically these involve four steps being 1) Analysis of current physical system, 2) Derivation of current logical system, 3) Specification of required logical system and 4) Specification of required physical system. SSADM (Structured Systems Analysis and Design Methodology) on the other hand is based primarily on three views being business system models, data structure models, and dynamic behaviour models. This approach consists of seven stages in the life cycle and a number of major techniques. A large amount literature exists about these types of approaches including the extensive work of authors like Edward Yourdon, Tom DeMarco, and Michael Jackson, however for the purposes of requirements elicitation alone, which is both very social and extremely dynamic, they can often be largely data oriented, overly constrictive, and too heavyweight.

2.5.5.2 Unified Modelling Language (UML)

Object Oriented (OO) approaches, such as the Rational Unified Process (RUP) (Kruchten 2003) and more specifically the Unified Modelling Language (UML) (OMG 2004), which can be used in cooperation with and as part of the RUP (Windle & Abreo 2002), is a widely adopted industry standard for the visual analysis and design of software systems. This approach includes established yet flexible notations such as Use Cases diagrams, Use Case descriptions, Class diagrams, Activity diagrams, Sequence diagrams, and State Transition diagrams, for use within the software development process. Of these, Use Cases are probably the most useful and widely used for requirements elicitation because of their simplicity and ability to record functional business processes. However UML like all modelling languages is more effective when used with stakeholders that are familiar with its formats, in the same way that OO approaches may be a natural way of looking at a system for developers, but necessarily for users. For the most part UML is used to model requirements once they have already been captured, and is as the name suggests more of a modelling technique rather than an elicitation approach on its own. As a result

UML is more effective within the requirements elicitation process when used with, and with the results from, other more traditional requirements elicitation techniques.

2.5.5.3 Soft Systems Methodology (SSM)

In the case of the Soft Systems Methodology (SSM) (Checkland & Scholes 1990), requirements elicitation is a defined but closely integrated activity within other aspects of the software development process. SSM concentrates on addressing organisational problems and dynamic change, where the focus is on addressing people issues as they relate to the development of the ideal system. The approach consists of seven stages being 1) The problem situation unstructured, 2) The problem situation structured, 3) Root definitions of relevant systems, 4) Conceptual models, 5) Comparison of stage 4 and stage 2, 6) Identify feasible and desirable changes, and 7) Action to improve the problem situation. The idea is that these stages can be performed iteratively with an experienced facilitator, ensuring that all the different viewpoints are explored, and unnecessary assumptions are not made. Concepts are represented using informal rich picture formats to make them easy to understand for the users, and promote the discovery of imaginative solutions. In general SSM is more useful at the very beginning of a project to explore the problems more so than the requirements, as it is one of the few approaches that is good at dealing with complex issues when the real objectives are still unclear. The SSM approach is however quite heavyweight and time consuming. Furthermore the process offers limited support for the design of completely new systems and is better suited to environments with existing systems.

2.5.5.4 Quality Functional Deployment (QFD)

Quality Functional Deployment (QFD) (Akao 1995), with the catchcry of “hearing the voice of the customer”, focuses on achieving customer satisfaction through quality based development. Using an integrated matrix and visual model known as ‘the House of Quality’, the approach is used to determine and design prioritized product development characteristics that consider and combine business priorities and technical requirements with customer needs, preferences, and expectations. The

objective is to produce a highly aligned system resulting in increased product acceptance, supported by benchmarked target values linked to metric based objectives for the project. Benefits of this approach include an increased coverage during the requirements elicitation process, and the early identification of key product features which can be used to support the design, development, and testing phases. Without the need for much formalism, QFD enables the iterative yet concurrent collection of both problem domain knowledge and solution domain knowledge. QFD has been used very successfully in many projects, although these for the most part have tended to be conducted in highly technical industrial organisations. Despite this the core QFD methodology has also been used within the requirements elicitation process to specifically address software quality characteristics.

2.5.5.5 Agile Methods

Agile Methods for the most part enforce very little upfront requirements elicitation but instead advocate incremental and iterative discovery throughout and integrated with the software development lifecycle (Martin, R. C. 2003). In addition to interviews and prototypes, Agile methods support the use of Customer or User Stories. These provide basic descriptions of the business processes and what the system needs to do to support them. Typically these are written on index cards by the customer and used as starting points for the development process. Additional requirements elicited as a result of the process from the ever-present customer are added to a Product Backlog, which represents a living requirements document consisting of prioritized system features and functions. Agile methods also encourage the use of prototypes, and particularly evolutionary ones (early and incomplete versions of the actual system).

2.5.6 Social Approaches

Social approaches are those that focus on the role humans play in not only the process of requirements elicitation, but also in designing and using the resultant system. Many of these could also be considered as collaborative approaches, because much like

collaborative approaches, social approaches typically involve groups of stakeholders working together to determine their needs.

2.5.6.1 User Centred Design (UCD)

User Centred Design (UCD) (Gulliksen, Lantz & Boivie 1999) is a widely accepted methodology for designing and producing software that actually meets the needs of the users. By actively involving the users as part of a multidisciplinary project team, UCD aims to ensure a unity between the technical system and the social system. In addition to the steps of developing user ownership, identifying user values, and audience definition, the requirements elicitation stage involves the analysis of users and their tasks via a range of techniques including observation, task analysis, and the iterative refinement of prototypes based on user feedback and evaluation by observation. Participatory Design (PD) (Carmel, Whitaker & George 1993; Chin, Rosson & Carroll 1997) is a type of UCD where the users are involved to an even greater degree throughout the entire software development process, and are responsible for many of the design decisions. A major advantage of PD is that it helps participants and especially the users take a personal stake in the project and system, and therefore they are more likely to work towards making it succeed (Moore & Shipman 2000). Another type of UCD is Contextual Design (CD) (Beyer, H. & Holtzblatt 1997), which begins with a contextual inquiry where the analyst interviews users and attempts to understand the way they work. CD also provides steps for building models and determining the physical and social environment for the system. This overview information is then used to design the user interfaces with even more user input. In general UCD focuses on the development of new applications where the users have high levels of ownership and less distance to the developers. As the name would imply, the process of UCD is mostly design driven rather than requirements driven, and is based on general principles rather than a guide consisting of a set of specific and simple clear cut steps.

2.5.6.3 ETHICS

The ETHICS (Effective Technical and Human Implementation of Computer-based Systems) approach (Mumford 1995) is derived from Socio-Technical Systems Design theory. Like other social approaches, ETHICS relies heavily on end-user participation during systems development. In fact the ETHICS approach consists of twelve main steps for the development and implementation of new systems, the first six of which relate in one way or another to the process of requirements elicitation. ETHICS expands the concept of traditional system requirements to include specific human requirements such as job satisfaction and quality of life. A variant of the ETHICS approach is QUICKEthics, which is a front-end process that requires a mix of activities directed at eliciting accurate information. These activities include questionnaires, group discussions, and information prioritization in order to develop an organised knowledge model of the real needs. The premise of the ETHICS approach is that for a system to be effective, the technology must fit closely with social and organisational factors. In particular this means that an improved quality of working life and enhanced job satisfaction for the users must be a major objective of the systems design process. One of the benefits claimed from using the ETHICS approach is that the resultant systems fit well and easily within the target organisation, however one of the main criticisms is that it is heavily dependent on management support and flexibility in organisational change.

2.5.6.3 WinWin

The WinWin approach (Boehm, B. et al. 1994) is based on a spiral process model, and involves at first each stakeholder capturing his or her desired objectives called “win-conditions”. Next the stakeholders detect the conflicts between their win-conditions and the individual requirement specifications for the system. They then try to find the agreement conditions that satisfy each stakeholder’s win-conditions. The process to resolve the conflicts and to find the agreements is formalized through a state transition diagram and typically guided by an experienced analyst. In this way the WinWin approach promotes group decision-making by managing the artefacts produced and used during the negotiation process, resulting in a proposed system

specification containing non-conflicting and compromised goals. As a result this approach addresses the development of a system specification more as a process of negotiating individual objectives and resolving conflicts, rather than a collaborative requirements elicitation activity. However the structure provided is very person-oriented and does focus on the issues of participants to increase cooperation and collect rationale.

2.5.7 Section Discussion

Many of the current approaches tend not be adopted for various reasons, but largely it is due to the level of support needed for their implementation. Many are not suitable for novice analysts, as they require a significant level of experience and expertise to be used effectively. It is also arguable that many of the available approaches are not sufficiently useful or practical, and the transfer of knowledge required to introduce these methods to industry is too difficult. We have seen that several of the existing approaches tend to be limited by the type of system, the stage of the project, or the domain, and are typically focused on the quality of requirements and not on other metrics such as speed and effort. It has been seen that most of the reviewed approaches, even excluding those we have classified as model-based approaches, depend on some underlying model of the system or the domain. Therefore it would appear that requirements elicitation approaches in general blur the line between elicitation and analysis, as opposed to a specific focus on the former.

We have seen that much like the techniques reviewed in the previous section, the approaches presented address the specific process of requirements elicitation to different degrees, and with different strengths and weaknesses. However in general it would appear that the Modelling, Combinational, and Collaborative approaches in general offer greater levels of detailed support for the elicitation of software system requirements. Most of the modelling approaches used for requirements elicitation have been developed specifically for this process, and are therefore well suited to uncovering the types of information needed at this stage in software development. In the case of combinational approaches, results have shown these approaches to be very effective in eliciting requirements, and greater than just the sum of the individual

techniques. In some cases such as when prototypes are operated by users under the observation of the analyst, the combination of these techniques has the potential to provide much richer and more detailed requirements information on both the business processes and the needs of the users. Collaborative approaches are a common and often default approach for requirements elicitation, and this is not surprising given that requirements elicitation by nature is a collaborative activity as it is rare that the goals and requirements for a system come from only one source or stakeholders. Multiple users provide a broader experience base (more heads on the problems means more ideas), and information sharing between users during collaboration can be very beneficial in improving the quality of requirements produced (Hickey, Dean & Nunamaker 1999). Well-structured collaborative group work such as JAD for example has regularly but unofficially reported productivity gains of anywhere from 10% to 70%.

We can conclude that an approach should be collaborative, combinational, specific, supportive, flexible, useable, and useful, with the right amount of structure and rigor. It should help breakdown any communication barriers between the analyst and the other stakeholders to allow meaningful discussion of the problems that need to be addressed and the subsequent requirements (Gambhir 2001). What appears to be missing from the reviewed approaches is one aimed at simplifying and optimizing the process through the seamless integration of techniques, guidelines, and tools (Lecoeuche, Mellish & Robertson 1998). Maiden and Rugg (Maiden, N. A. M. & Rugg 1996) identified this need due to the lack of guidance in planning a systematic, well-grounded acquisition programme. They also acknowledge that to provide a more holistic approach, a range of acquisition methods needs to be used, which recognises the complexities and specifics of software requirements elicitation. Christel and Kang (Christel & Kang 1992) in their seminal work on requirements elicitation issues advocate “a better approach to requirements elicitation is to synthesize the various methods and techniques into a methodology, which then can be instantiated based upon a target system’s attribute”, meaning a situational approach which can be engineered with construction guidelines. Therefore any new approach should be able to be used in cooperation with other techniques, approaches, and tools, and not exclude them. It is also precisely because of the many and varied contexts in which software development is performed, and the large number of factors, techniques, and

issues that may have an affect and are involved even in the most typical of processes, that is not only suitable, but essential for approaches to be situational.

In summary, the key points identified in this section are:

1. Modelling approaches are generally shallow and domain-centric

2. Combinational approaches can be restrictive and require significant experience

3. Collaborative approaches can be very involved and require significant expertise

4. Methodological approaches are suitable for developers but not necessary users

5. Social approaches are high-level and heavily dependent on the users

6. There is a need for simple but focused situational approaches

2.6 Requirements Elicitation Tools

2.6.1 Section Overview

A ‘tool’ is an implement, such as software or an artefact, used in practice to accomplish some act, in this case being requirements elicitation. Many tools have been developed for RE (see (Atlantic Systems Guild 2007), (Alexander, I. 2007), and (INCOSE 2007) for surveys), however these largely concentrate on modelling and management, rather than being explicitly concerned with elicitation. This is mainly because modelling and management has often been the focus of RE tool development, since elicitation is often considered as a soft or social activity, and not necessarily conducive to tool support. Despite this, there is significant number of tools that are used during requirements elicitation, and we review some of them below in order to evaluate their relative strengths and weaknesses, and identify those areas of requirements elicitation tool support that are in need of further attention.

As with techniques and approaches, there is no universally accepted way to classify requirement elicitation tools. As a result, the classification of requirements elicitation tools could be based on their architecture, on the specific task they support, or on the underlying method they embody. All of these are valid, however, we have elected to group our selection of tools based on the way in which they are used within the context of the requirements elicitation process. Many of the specific tools reviewed can be placed in more than one of the classes we have used, but the focus of this classification is more on providing a representative sample of the spectrum of available tools rather than creating a definitive hierarchy.

2.6.2 Basic Tools

Basic tools represent some of the more general tools used during requirements elicitation without being associated with any particular process, technique, or approach. We have elected not to include generic tools such as word processors and spreadsheets as this would be dispersing the scope of our review too far.

2.6.2.1 Template Tools

The Volere Requirements Specification template (Atlantic Systems Guild 2003) is really an example of a software requirements documentation tool. Another is the IEEE Std 830-1998 Software Requirements Specification (IEEE 1998a). These templates represent the most basic type of tool used by analysts to support the process of requirements elicitation. Templates can be used not only as a guide to document requirements, but also as a basis for structuring discussions about requirements with stakeholders. More detailed specification templates and patterns for requirements such as those proposed in (Durán Toro et al. 1999) can help analysts and users elicit, express, and record requirements information using natural language. These are similar in their approach and usage as Cockburn's Use Case template (Cockburn 2001), and the Volere Requirement Shell (Robertson, S. & Robertson 1999). Although all of these tools provide a relatively easy way of representing requirements and requirements related information, they lack any sort of interactivity with the actual process of elicitation, and are effective only as high-level checklists or guides for the types of information that should be elicited.

2.6.2.2 Management Tools

RequisitePro (IBM 2005) represents only one of a number of commercially available tools primarily designed for requirements management. These tools provide format-based support for the recording of requirements, as well as a means for identification and organisation of requirements within a hierarchy, traceability of requirements from their source and their relationships with other requirements, as well as change and version control. Examples of other requirements management tools that provide the same basic functionality include CaliberRM (Borland 2005), RMTrak (RMTrak 2005), RTM (Serena 2005), and TRUEreq (Truereq 2005). All of these provide the same core features of any requirements management tool, however, there are some minor differences that can provide additional elicitation assistance through operations such as gap analysis and market analysis, although this falls short of being technique support or process guidelines. Active! Focus (Falafel Software 2005) is slightly

different in that it supports full application lifecycle management include project planning and defect tracking system closely integrated with requirements management. DOORS (Telelogic 2005), also goes beyond the typical functionalities of requirements management tools by providing more advanced modelling and analysis capabilities. Despite these differences it can be said that the available requirements management tools included all those listed above offer mechanisms only for the capture of requirements, but supply limited if any support for their elicitation from stakeholders in terms of features and instructions.

2.6.2.3 Diagramming Tools

Diagramming tools such as Visio (Microsoft 2005) and FlowCharter (Corel 2005) can also be used during requirements elicitation. These enable the analyst to work with the stakeholders to draw informal models of the domains, the system, and existing processes using simple and understandable graphical representations. The two examples given and others like them do have available symbol palettes and plug-ins to support specific modelling notation such as UML or SAD, and as a result they can be used for mapping processes and objects related to the system and tasks. In general, these types of tools are very flexible, simple to use, and can produce results that are easy to understand, with very minimal training required for both the analyst and the stakeholders (Satterfield 2006). However, once again they do not provide any sort of formal guidance for the process of requirements elicitation, nor do they facilitate in the interaction between the analyst and stakeholders.

2.6.2.4 Survey Tools

QuestionPro (QuestionPro 2005) just like The Survey System (Creative Research Systems 2005) are both examples of generic tools for building online questionnaires and surveys. Often these tools are touted as being computer-based interviewing, and as a result are sometimes referred to as automating the requirements elicitation process (e.g. in (Kassel & Malloy 2003)). In (Hands, Peiris & Gregor 2004) a web-based interviewing tool for requirements elicitation is presented, but by their own admission this is only intended as a precursor to an actual face-to-face interview

between users and developers/analysts, in order to collect starting point information. The main motivation for this and other tools like it is that a computer presents the image of an infinitely patient and non-threatening, non-judgemental interviewer as opposed to a human. These systems still require the analyst to develop explicitly the set of questions required to elicit the information, and the tool merely presents them and enforces selected rules for their completion. These types of system can only provide shallow information and are entirely dependent on the quality of the questions inputted and therefore heavily reliant on the skills of the analysts, with very little value added other than being able to generate some very simple statistical information about the responses. Essentially no specific process support is provided, and what is actually offered is a generic tool for asking questions to people and getting their responses via a computer. A similar prototype example, in concept at least, is described in (Kassel & Malloy 2003), which claims to partially automate requirements elicitation and specification processes. The analyst or domain expert creates questionnaires with close-ended questions using a prescribed XML schema, and these are loaded by the interface, which enables the customers and users to answer them via the Internet. Once again this is just a questionnaire creation and completion tool, however it does provide the facilities to export the results as a basic requirements specification.

2.6.3 Method Tools

Method tools are those that support and enact a specific requirements elicitation technique or approach, and subsequently represent by far the largest category of all requirements elicitation tools. In fact as we will see, all requirements elicitation tools to some degree are related to one or more elicitation technique or approach. Conversely most requirement elicitation techniques and approaches have at least one available tool that can be used to support their execution.

2.6.3.1 Goal-based Tools

A number of tools have been developed to support the various goal-based approaches used for requirements elicitation. Objectiver (formally GRAIL) (Cediti 2005) is one

of those specifically supporting a goal-oriented methodology for RE, and more specifically the KAOS meta-model (Dardenne, van Lamsweerde & Fickas 1993). Tools also exist for the other goal-driven approaches to requirements elicitation including OME (Organization Modelling Environment) (University of Toronto 2006) for the i* framework (Yu 1997) as part of the larger TROPOS project (Tropos 2006), EasyWinWin (Briggs & Grünbacher 2002) for the WinWin (Boehm, B. et al. 1998) negotiation model approach, and a tool reported in (Tuunanen & Rossi 2004) for Critical Success Chains. For the most part these tools are based once again on the graphical generation and representation of models. The obvious limitation of using one of these tools is that by definition they require the underlying goal modelling method to be adopted for the process of requirements elicitation as well.

2.6.3.2 Modeling Tools

In many cases specific modelling tools are utilized to assist the process of requirements elicitation. One such example is ArgoUML (Tigris 2005) (open source but made commercially available by Gentleware as 'Poseidon for UML'), which allows analyst and stakeholders to quickly develop UML models including Use Case diagrams and descriptions. In fact, for each modelling technique applicable to Software Engineering, there is often a range of supporting tools. UML alone has literally dozens of available tools including Rational Rose (IBM 2006) and Profesy (Sofea 2005). Often these tools are integrated within CASE tools and larger lifecycle management platforms, helping put the modelling process and models generated into the context of the software development process. For the most part, these tools offer little if any additional support for the actual process of requirements elicitation other than those basic guidelines and assistance offered when creating models, such as the Design Critics and Checklists provided in ArgoUML. How the requirements information is elicited to build those models is not addressed, and although the models generated by using these tools typically do conform to some standard, they still suffer from much the same inadequacies as the more general diagramming tools with respect to being able to support the process.

2.6.3.3 Graphical User Interface (GUI) Tools

The Graphical Requirements Collector (GRC) (Moore & Shipman 2000) is an example of a requirements elicitation tool based on the specific technique of prototyping. Using this tool, probable end users sit down and make their own application by creating mock screens and dropping widgets into those screens. Users are also able to annotate their actions by providing argumentation about each widget and screen produced. This argumentation provides the primary means for obtaining requirements. Along the same lines, a requirements elicitation tool is presented in (Shimakage & Hazeyama 2004) where the developers and requesters design screen images by using a Graphical User Interface (GUI) image editor as part of a high-level requirements elicitation process that including Use Cases. As uncovered from experiments using the GRC tool, a lot of procedural information is collected, but the prototypes constructed by the users seldom go beyond existing functionality, and original ideas for solving the problems were not produced (Moore & Shipman 2000). This suggests that tools based on users creating screens and GUIs are restrictive in that they predetermine a solution space by specifying much of the design rather than the focusing on the requirements. Furthermore these types of tools assume that the users are familiar with computer applications that run in similar types of environments.

2.6.3.4 Scenario-based Tools

The ART-SCENE's (City University 2005) Scenario Presenter is an interactive tool for discovering, acquiring, and describing requirements for new systems using scenarios. It has two main parts being 1) a scenario generation tool that automatically generates normal and alternative scenario courses from a Use Case specification, and 2) a walkthrough tool for systematically discovering and documenting requirements from the scenarios. The walkthrough component enables a step-by-step review of normal and alternative course events, editing and adding comments to scenarios, and automatically tracing new requirements to scenario events. A number of scenario-based tools also came out of the CREWS project (CREWS 1999) including CREWS-SAVRE (Sutcliffe, A. G. et al. 1998) which provides Use Case and scenario editing

tools, a scenario generation tool, and semi-automatic validation of incomplete and incorrect system requirements using commonly occurring scenario event patterns, and PRIME-CREWS (Haumer, Pohl & Weidenhaupt 1998) which supports the use of scenarios, that are instances of Use Cases recorded in the form of real-world scenes in the construction and validation of goal models. A positive aspect of these tools is that they are based on a sound underlying method, which although relatively lightweight, is specific to requirements elicitation. However scenario-based tools assume not only that the stakeholders are familiar with the processes the system must support, but that those processes are able to be clearly defined at the time of elicitation.

2.6.3.5 Knowledge Acquisition Tools

WebGrid (Shaw & Gaines 1995) as an example of a knowledge acquisition support tool is a web-based implementation of George Kelly's Repertory Grid technique for building conceptual models based on Personal Construct Psychology (PCP). Along the same lines WebMap (Gaines & Shaw 1995) is also an online knowledge acquisition tool, but for the creation of Concept Maps. Shaw and Gaines (Shaw & Gaines 1996) state that these types of tools, that are designed to manage and organise the large amounts of heterogeneous data gathered in the early phases of Knowledge Engineering particularly from experts, are applicable to the management and organisation of similar data collected in the early phases of requirements elicitation. Although we agree that they can certainly be applied, their usage is limited by the lack of specific support offered by them for the process of requirements elicitation.

2.6.4 Cognitive Tools

Over the years several tools have been developed with cognitive support for the requirements elicitation analyst in mind. In most cases these have relied on techniques from the fields of Natural Language Processing (NLP) and Artificial Intelligence (AI). Although not specifically for elicitation, the idea of using these techniques for RE tools was addressed by the NATURE prototype (Pohl et al. 1994), but only at a very preliminary and high-level proposal stage. However the development of cognitive

tools to support requirements elicitation has been a subject that has attracted some degree of attention from a variety of other sources.

2.6.4.1 The Requirements Apprentice

The Requirements Apprentice (Reubenstein & Waters 1991) (as part of the Programmer's Apprentice project), despite claiming to be for the acquisition of requirements, is in fact concerned with and focused on the transition between informal and formal specifications. The Requirements Apprentice relies on a variety of techniques, including dependency-direct reasoning, hybrid knowledge representation, and the reuse of common forms (clichés), and specific structures of the domain to infer additional information. As far as elicitation is concerned, the Requirements Apprentice is really only intended to be used for the formalisation and validation of requirements. This is achieved using Common Lisp rules and input based on a structured language of high-level words to enable flexible and complex reasoning. Although proposed as only a research prototype of an intelligent assistant, the Requirements Apprentice can only be used by an experience and trained analyst. The tool does not interact directly with the end user, and so the analyst is still ultimately responsible for communicating with the end-users and entering the requirements related information. Put simply, although the Requirements Apprentice works to improve the quality of a set of requirements, it still relies on existing methods of acquisition, such as questionnaires, interviews, or workshops to elicit the initial requirements (Moore & Shipman 2000).

2.6.4.2 ACME/PRIME

ACME/PRIME (A Conceptual Modelling Environment/PRocess Implementation MEthodology) (Feblowitz et al. 1996) represents a methodology and tool for resolving the under or over specification of service-oriented systems, after the initial requirements have already been elicited. This tool is primarily concerned with modelling business processes for simulation, and refining what has in reality already been elicited. The methodology and therefore the tool does address the analysis of business processes, but it is fixed to a particular business process-centred approach,

and therefore a specific type of project and system. ACME/PRIME does not interact directly with the users and is not web-based. Because of its use of graphical process models (maps) and a semantic action language (dialogs), a significant amount of training and expertise is required for the analyst to use the method and tool effectively. Although the tool supports the method and subsequently the process, it does not drive it.

2.6.4.3 KBRA

In the Knowledge-Based Requirements Assistant (KBRA) (Czuchry & Harris 1988) system like many tools apparently for requirements elicitation, the analyst is required to enter in the information that has already been elicited, but in either textual note form or as context and state diagrams. In this way the tool is more like a notebook manager for the analyst to support analysis and modelling more than anything else. KBRA has a reusable requirements library which provides much of the functionality including critiquing and completing the informal and evolving system descriptions, and some basic AI capabilities which facilitate property inheritance, automatic classification, and constraint propagation. At best this tool will help the analyst identify what remains to be elicited or what has been elicited that may not be correct.

2.6.4.4 AbstFinder

AbstFinder (Goldin & Berry 1994) offers an approach and tool for identifying abstractions in natural language text already collected from customers and users. Once again this represents a system supporting the analysis of requirements more so than the elicitation of them. Designed to help the requirements analyst massage transcripts of interviews into quality statements of what is needed, this tool is based on techniques of natural language processing including the identification of repeated phrases and lexical affinities. The analyst must review all the raw information first, and is still required to actually produce the requirements statements, however there is some assurance that no major item or topic will be overlooked thanks to the results produced by the tool. When evaluated against both two similar tools and three human

expert requirements analysts, the AbstFinder was found to provide some measure of improvement with respect to requirements quality.

2.6.4.5 KBRAS

The Knowledge-Base Requirements Acquisition System (KBRAS) (Zeroual 1991) uses a LISP/PROLOG environment for reasoning in the same way as KBRA. Elicitation is driven primary by a conceptual framework that consists of several models of the target system (environmental, functional, and behavioural). KBRAS uses a restricted natural language and graphics interface to identify objects in the domain, describe attributes, detailed constraints and relationships, and express goals and tasks. A questioning dialogue drives this process between the user and the system based on reasoning via an inference mechanism as more information is entered. This same mechanism also allows for some completeness and consistency checking. As a result, KBRAS is one of the very few tools that not only actually supports the elicitation of requirements, or at least domain information, but also automates the process to some degree. However the interaction is once more only with an experience analyst or domain expert, and the slow response time of the tool meant that it was not evaluated in the real world.

2.6.4.6 RECAP

The starting point for the Requirements Elicitation, Capture and Analysis Process prototype tool (RECAP) (Edwards et al. 1995) is once again the input of informal text containing requirements information, assumed to have already been elicited from the customers and users. The text is first filtered and then parsed by the tool for grammatical patterns and relevant subjects, and relevant sections are then ‘tagged’ and formatted semi-automatically into a requirements template, thus producing a set of semi-formal requirements. A Boolean (Logic) based language is then used by the analyst to convert the identified and indexed requirements into domain rules, which can be analysed against the other requirements relevant to that domain subject. This is once again an example of a text analysis tool, which may help identify potential requirements from a requirements document, but does not actually support directly the

elicitation of requirements from users and customers. RECAP was not formally evaluated and has only been reported at a very early stage in its development.

2.6.4.7 FRED

The FRED (First Requirements Elucidator Demonstration) tool (Kasser 2004) attempts to provide at least a partial solution to the problem of poorly written requirements by combining the principles of Total Quality Management, expert systems, and knowledge management. This tool performs syntactic processing on textual requirements and notifies the user when characteristics of poorly written requirements are present. It is then up to the users to determine if a defect actually exists, and take the necessary corrective action. FRED is not able to comment on the completeness of the requirements set, or on any conflicts between requirements. Once again this is a case of tool focused on requirements analysis rather than elicitation, as it is assumed that the original set of requirements have already been elicited and documented by the analyst or someone else.

2.6.5 Platform Tools

Sometimes referred to as blackboards or environments, platform tools for requirements elicitation provide a range of features, many of which are similar to more generic groupware applications. Some early attempts were made to develop integrated environments for requirement elicitation including (Palmer & Fields 1992), however most of these have consisted of two major subsystems being one to control meeting processes and tools, and another to control information and methods. Although these ideas were certainly promising and a step in the right direction, very few were ever actually implemented and evaluated, and subsequently details on their operation are limited.

2.6.5.1 AMORE

The Advanced Multimedia Organizer for Requirements Elicitation (AMORE) (Christel, Wood & Stevens 1993; Wood, D. P., Christel & Stevens 1994) is used to

"store requirements in as close to their natural forms as possible to maximize traceability and to promote understanding of original intentions and motivations". This tool was intended to fill the gap between raw sources of requirements and CASE tools, and is focused on "capturing and organizing the information generated during requirements elicitation" rather than the process of requirements elicitation itself. Within AMORE, requirements are represented in a parent/child modifiable hierarchy model, which can be navigated through by the user. The basic unit of information is the requirement, which has a number of specified attributes such as rationale that can be supported by attachments such as video or audio clips and other documents and files. Unfortunately AMORE was not fully realised with the process guidance and intelligent aspects never being implemented.

2.6.5.2 CRETA

CRETA (Collaborative Requirements Engineering Support Tool) (Togneri, de Almeida Falbo & de Menezes 2002) aims at supporting the main activities of the RE process, and is intended to be used by not just the analyst but the domain experts and project team also. This web-based application enables users to record appointments, exchange email, and includes other features such as a group calendar, forums, and chat. CRETA is able to store and retrieve documents, setup meetings, and build questionnaires, the last being the only real support for elicitation. This tool is supposed to be used almost exclusively asynchronously and in reality is more of a content management tool for storing general but project related information. The functionality of CRETA is therefore predominately administrative, and its implementation to date has not been evaluated.

2.6.5.3 WRET

The WRET (Web-based Requirements Elicitation Tool) prototype (Hassan & Salim 2004) is a web-based tool built on the Viewpoint approach using the Lotus Notes platform specifically for distributed stakeholders. Information is captured and stored in three basic templates from the underlying approach (Viewpoint, Service, and Concern) in a repository with additional user and project administration data.

Although mainly a requirements management tool, WRET provides a small static list of activities specific to the enforced Viewpoint approach, some of which are requirements elicitation related. A very brief and informal evaluation of this tool was performed, however only general observations were presented with no basis for comparison or empirical evidence. Once again WRET represents more of an information repository for requirements elicitation, rather than a tool that supports the actual process, despite having a limited implementation of the Viewpoint approach.

2.6.5.4 ADREAM

The ADREAM (Agent-assisted Distributed Requirements Elicitation And Management) tool (Chang, Krishna & Ghose 2003) is an initial attempt at using an agent-mediated architecture to elicit requirements directly from distributed stakeholders. Based on a common requirements repository, each stakeholder user is represented by a stakeholder agent. Elicitation of the initial requirements is primarily driven by a predetermined ontology for the specific project and system (i.e. a list of relevant concepts). By displaying the requirements entered or updated by other stakeholders to each user, this is intended to further encourage additional elicitation to take place. The users may also change the ontology, which is also presented to the other stakeholders in the hope of again eliciting more requirements. The intelligence of the agents is limited, and based on the identification of conflicts, the association of goals to requirements, and the formalization of natural language. This web-based application, which is both portable and distributed, is not described in detail and still under development.

2.6.5.5 RETH

RETH (Requirements Engineering Through Hypertext) (Kaindl 2004; Kaindl, Kramer & Hailing 2001) is both a method and a supporting tool focused on modelling functional requirements, scenarios, and goals, as well as their relations to each other. Based on a RE meta-model of objects and relationships, RETH provides a process guide that describes the use of the tool according to the process defined by the method, in addition to basic implementations of a hyperlink generator, an

association/relations generator, and a meta-model checker. RETH includes some basic requirements management functions like versioning, traceability, and export. The tool does provide step-by-step instructions for entering in the information, as well as advice for users performing the task, and is targeted at novice users. The look and feel of this application is purposely intended to be similar to the Microsoft Office suite of tools to promote ease of adoption and use. RETH was first evaluated in practice via an informal useability test, and then a more formal useability experiment where it was found that the tool with the guidance enabled users to perform the tasks faster. The tool is intended to be used by individual analysts as opposed to groups of stakeholders.

2.6.6 Collaborative Tools

Collaborative tools represent a wide range of support applications that have been applied to requirements elicitation. This covers everything from basic support tools such as discussion boards, video conferencing, and idea capture software, all the way through to virtual environments specifically designed for group projects. It should come as no surprise that there is a large number of tools that claim to support collaborative requirements elicitation considering that Software Engineering at any level is a collaborative activity, and almost always involves the combined and coordinated work of various people and processes (Saeki 1995). Collaborative tools aim to promote a shared understanding and joint problem solving.

2.6.6.1 GroupSystems

GroupSystems (GroupSystems 2005) was originally developed at the University of Arizona and is certainly one of the most popular and well-known groupware tools for Software Engineering, and one of the few tools in this space to become commercially successful. More specifically GroupSystems has been applied to requirements elicitation research in a number of cases including (Hickey, Dean & Nunamaker 1999) and (Hannola, Elfvengren & Tuominen 2005). The fundamental premise and aim of GroupSystems is to build consensus through process-based team collaboration via eMeetings, virtual meetings, or face-to-face sessions, through accelerated and

cooperative problem-solving and decision-making processes. The tool provides facilities for brainstorming, organising ideas, voting, and prioritizing, in addition to consensus building and action planning. It is claimed that by using GroupSystems, meeting time and costs can be decreased by more than 50%. Although it does contain a basic process model and a range of activities that be selectively applied for each step and is particularly suited to software product development, it is general for all types of meetings, and does not provide additional support specifically for eliciting requirements such as how to plan and run a workshop.

2.6.6.2 Hyper Minutes

Hyper Minutes is a tool reported in (Kaiya, Saeki & Ochimizu 1995) which used hypermedia to support requirements elicitation meetings. Although not based on any specific or prescribed elicitation process, this tool records, stores, and organises the contents of the meetings (audio, video, graphics, and text), generates minutes and agendas for future meetings based on the repository of records from a past meeting, and provides interfaces for the user to navigate through the repository records on or offline. The concept of 'hyper minutes' is both interesting and useful for requirements elicitation, however this is just another example of a generic tool for recording the contents of any meeting. No special assistance or guidance is provided for the process of requirements elicitation, like for example the ability to link recorded items to potential goals, requirements, and features of the target system.

2.6.6.3 Centra Live

Centra Live for eMeetings (formally Centra Symposium) (Centra 2005) provides facilities for the replication of same place meetings through a set of tools and capabilities including online audio and video conferencing, application sharing, file transfer, surveys (polling), feedback, whiteboards, and chat. Some additional management and coordination features are provided including a basic knowledge centre for the storage and retrieval of reference material and documents, and meeting scheduling facilities for automated set-up and follow-up. An example implementation of this tool to requirement elicitation can be found in (Lloyd, Rosson & Arthur 2002)

where it was used for real-time virtual meeting support, in conjunction with another tool called MOOsburg for file sharing and asynchronous discussions, in order to evaluate the effectiveness of elicitation techniques in distributed RE. Although this tool does allow for requirements elicitation meetings with stakeholders across different geographic locations, it does not guide the process of requirements elicitation, or provide any specific assistance for this activity in the form of direction or structure. As a result and as far as requirements elicitation is concerned, Centra Symposium offers the same level of support as several other commercially available tools including Microsoft NetMeeting and WebEx.

2.6.6.4 TeamWave Workplace

TeamWave Workplace (Roseman 2005; TeamWave 2005) is a web-based tool that supports asynchronous and synchronous collaborative activities for distributed groups with the aim of facilitating working together and building a community presence. Process guidance is limited to being able to create different 'rooms' for different types of activities, and linking them in some sequence or order. It is also possible to associate an array of different 'tools' to each room including a concept mapping tool, note taker, or document repository. The users can modify the contents of these rooms or workspaces at any time and at the same time, making it very flexible and dynamic. Each room is essentially created around a large persistent whiteboard, which forms the base for diagrams, text, and other representations. The standard set of tools that is provided offers support for file exchange, embedding of graphics, creating hyperlinks for navigation in, out, and around the rooms, text, audio, and video communications, message boards, post-it style commenting, storing and distributing information, and concept mapping. TeamWave Workplace has been applied by demonstration specifically for distributed RE by Herlea and Greenberg (Herela & Greenberg 1998). It is claimed that the interface of this tool is designed to be flexible and easy to use and that new users can learn to use Workplace in a matter of minutes. However to an inexperienced computer user it may appear complex and be conceptually difficult. Therefore TeamWave Workplace is not really appropriate for all stakeholders, and offers no real requirements elicitation specific support.

2.6.6.5 iBistro

iBistro (Braun, Bruegge & Dutoit 2001) is an experimental environment for capturing informal meetings using context-aware devices such as electronic whiteboards, video cameras, and location trackers. Meeting minutes are structured using a rationale-based approach, and these can be retrieved during subsequent informal meetings. The tool supports synchronous but not distributed meetings, and asynchronous activities including reviewing past meeting minutes and editing content. iBistro is essentially a time/event based multimedia meeting minutes management system, and is really focused on capturing detailed minutes of any type of meeting but especially brainstorming, rather than driving elicitation sessions per se. In fact one of the main goals of the iBistro development research was to not introduce a process for the meetings at all. The tool has not yet been evaluated or completely developed, although this is planned and areas of improvement have already been identified.

2.6.6.6 Compendium

Compendium (Compendium Institute 2005) (formally knows as QuestMap and gIBIS before that) is an Open Source commercially available product, providing software support for the IBIS (Issue-Based Information System) methodology. Using this tool, meetings can be mediated by Dialog Mapping, and design rationale can be captured in the form of picture and text based concept diagrams in a visual hypertext environment. Compendium also includes collaborative modelling and management of information in order to representing the group memory. This tool is also an example of the subclass of collaborative tools concerned with creativity, which covers tools that support other requirements elicitation activities such as mind mapping, concept mapping, and idea generation. Although these tools provide the mechanism for the externalisation of information and knowledge, it can hardly be said that they act as a catalyst or stimulant to the creative thinking process.

2.6.7 Section Discussion

We have seen that there exists a wide variety of tools that can be used to support the process of requirements elicitation. However only very few have attempted to address the actual task of eliciting software requirements directly with a tool, as was also identified in (Cucchiarelli, Panti & Valenti 1994). For the most part these tools were limited in scope and functionality, and required significant expertise to use them effectively. Most of the software-based tools we have reviewed are used mainly to facilitate communication during requirements elicitation, or to support a specific technique or approach.

One of the biggest issues we can identify is that many tools claim to aid requirements elicitation, when in actuality they support the simpler task of requirements capture as in the case of (Ko, Park & Seo 2000), or analysis/modelling as is the case for (de Freitas et al. 2003). The starting point for these tools is after the requirements have already been elicited, sometimes informally, rather than the actual elicitation activity itself. The vast majority of tools deal with what to do with the information once it has been elicited from the sources in order to represent it, store it electronically, organise and manage it, elicit more requirements, or improve its quality. Most provide little, if any, real guidance for the acquisition task. Furthermore many tools simply translate requirements, as opposed to actively extracting them through dialogue with the stakeholders and writing requirements through interaction (Lecoeuche, Mellish & Robertson 1998). Because some of the tools do not use natural language as the input format, or do not interact directly with stakeholders, they are not suitable for the early stages of requirements elicitation, and are heavily dependent on the internal abilities of an expert analyst.

As a result, the term requirements elicitation needs to be used more carefully especially with respect to tool support, as it would appear to be constantly misused in the literature. Of the much reduced number of tools that actually do provide active support for the specific process of requirement elicitation, most have still tended to be quite limited because they are either a) simple and generic or b) comprehensive but domain or method specific (Scott & Cook 2003). Another common criticism of many

tools and techniques for requirements elicitation is that the resultant information is not available or presented in a meaningful way that can be readily or easily used (Moore & Shipman 2000). Furthermore most current requirements elicitation tools are not integrated with other commonly used and available commercial tools used in Software Engineering for RE, design, and testing.

It seems that a lot has been promised with respect to developing tools with great functionality for requirements elicitation, but in reality most have only made it to the high-level design stage. In fact in 1996 it was identified that most tools for requirements elicitation were only in the development stage (Playle & Schroeder 1996), and unfortunately this still appears to be the case. Many tools reported in the literature have not made it past the initial proposals or prototypes. Those few that have reached some stage of physical implementation have almost without exception not been evaluated sufficiently to provide reliable empirical evidence of their value and worth. Therefore it is impossible to determine if in fact these tools would provide any improvement for requirements elicitation at all. Furthermore most of the tools have not been based on a sound theoretical foundation that would ensure that some part of the overall software development problem is addressed.

Calls for more and better tool support for requirements elicitation based on a variety of reasons and motivations can be found extensively throughout the available literature. Macaulay makes note of the fact that within the RE process, the use of automated tools is desirable (Macaulay, L. 1996). Nuseibeh and Easterbrook go even further by stating that “to enable effective management of an integrated RE process, automated tool support is essential” (Nuseibeh & Easterbrook 2000), a position echoed by Maiden and Sutcliffe (Maiden, N. A. M. & Sutcliffe 1993). In (Lecoeuche, Mellish & Robertson 1998) Lecoeuche, Mellish, and Robertson state, “the elicitation process is a complex task which necessitates computer support. Elicitation systems should ideally help their users check the correctness of the specifications obtained but also actively guide them in the acquisition of the requirements”. Likewise Rolland and Prakash say that, “since Requirements Engineering is a complex task, advice/guidance on which activities are appropriate in given situations as well as on how these activities are to be performed must be provided … [but] considerable freedom in deciding which activity is to be done next must be made available to the

requirements engineer" (Rolland & Prakash 2000). Additionally it has been identified that there is still a real need for collaborative tools to support the key aspects of the elicitation process (Hickey, Dean & Nunamaker 1999).

Process guidance was one of the five areas of RE identified by the NATURE prototype (Pohl et al. 1994) as being able to take advantage of artificial intelligence (AI) tool support. The other four were knowledge representation, reverse engineering, specification reuse, and finally domain abstractions. Hickey, Dean, and Nunamaker (Hickey, Dean & Nunamaker 1999) proposed with respect to tool support for requirements elicitation that (1) a more specific process may help completeness problems, (2) more specific forms and templates that prompt users for specific information may also help users focus and provide more details, and (3) more active facilitation is required in terms of feedback and guidance. Rolland and Prakash say "tool support has been lacking in two main directions by a) providing process support, and (b) adapting to the needs of specific systems" (Rolland & Prakash 2000). New web-based and interactive technologies mean that an excellent opportunity now exists to incorporate all the required aspects of good requirements elicitation tool support with existing methods such as the integration of intelligence for process guidance, task automation, and cognitive assistance for the analyst and stakeholders.

In summary, the key points identified in this section are:

1. Basic tools provide only high-level capture and representation support

2. Method tools enforce a specific techniques in a particular way

3. Cognitive tools require expert users and have not been evaluated

4. Platform tools are primarily administrative and only partially implemented

5. Collaborative tools support only generic Software Engineering processes

6. Most tools do not actually provide any specific process guidance

2.7 Chapter Summary

Throughout this chapter and in Section 2.2 in particular, we have seen that RE, and more specifically requirements elicitation, has now been identified as a separate, distinct, and important field within the discipline of Software Engineering. But this area is relatively new even for Computer Science, with real momentum only occurring in the late 1980's through dedicated conferences, symposiums and workshops. However significant advancements in the field of RE have been made over the past two decades given the short timeframe and the number of people working in the area. With respect to requirements elicitation, most of the work has been directed towards improving this complex process through the development and implementation of various techniques, approaches, and tools. Despite this we can still say that requirements elicitation has received insufficient attention in research to date, especially when compared to mature software engineering areas such as testing and coding, and many issues remain open and ripe for investigation.

We saw in Section 2.3 that although a number of different process models have been proposed over the years as generic roadmaps to address the elicitation of requirements for software systems, there still remains a lack of appropriately flexible guidelines and sufficiently detailed steps, which can be used by the majority of practitioners in typical projects. In fact both the quantity and quality of sufficiently detailed process guidelines is very limited, especially with respect to technique selection and addressing the contextual factors of requirements elicitation. Subsequently there is a gap that needs to be filled between the two extremes of high-level generic guidelines, and detailed instructions for specific techniques, to provide analysts with the required level of situational process support. Addressing this issue is made difficult by the lack of empirical research, case studies, and experience reports in the literature on the specific topic of requirements elicitation processes, and the effect of contextual factors on how it is performed.

From our review of elicitation techniques in Section 2.4, we can say that no one technique is sufficient for the process, and that a combination of techniques is actually required for successful requirements elicitation. For the most part this is due to each

technique having its own relative strengths and weakness, and the fact that certain techniques are more effective for particular types of situations, participants, and information. Many of these techniques have been borrowed straight from other disciplines as generic methods, some have been adapted for use in requirements elicitation, and only a few have been designed specifically for the process of requirements elicitation. In the majority of projects several of these methods are typically employed at different stages in the software development life cycle in cooperation where complementary. What is missing from most of these techniques, however, is practical support for the implementation and integration with a structured and rigorous requirements elicitation process.

In Section 2.5 we were able to identify a number of useful and useable approaches to requirements elicitation, however much like the requirements elicitation techniques reviewed, each type of approach was only appropriate under certain types of conditions. However unlike techniques, requirements elicitation approaches all require significant commitment from both the analyst and stakeholders. Of those examined, combinational and collaborative approaches supported by facilitated group work seem to provide the most requirements elicitation process specific support, and allow for the employment of one or more integrated techniques. What is really needed are focused requirements elicitation approaches that are both easy to use and useful to analysts, and which can be tailored dynamically depending on the situation.

The tools we surveyed in Section 2.6 all claim to support requirements elicitation to some extent, however, what they almost all lack is an accompanying process or framework in which to use them. Although this is done deliberately in some cases to make them more adaptable and flexible to multiple contexts, it does not provide novices, especially those working without a mature, prescribed, or defined software development process, the necessary guidelines for their implementation or the motivation for their use. Collaborative tools not only support workshops, which we have determined to be effective, but also provide new dimensions to the process. We have seen that many important areas remain open for investigation with respect to providing tool support for requirements elicitation including intelligent assistance for novice analysts and direct interaction with the system stakeholders without the use of a semi-formal modelling or analysis technique. To date most tools have been fairly

limited in their application, and have required a high level of experience and expertise. For the most part the existing tools have failed to address the complexities and labour-intensiveness of the requirements elicitation process.

In summary, the number of different factors that must be taken into consideration when performing requirements elicitation prohibits a single definitive technique, approach, or tool for all projects and systems. The experience and expertise of the analyst, time and cost constraints, volatility of the scope, and the context in which the project is conducted all have significant influence on the way in which the process should be performed. Despite the large number of available methods, and several major efforts to develop frameworks and guidelines, requirements elicitation still remains more of an art than a science. This can also be partly attributed to the lack of method evaluation and comparison that has been performed under strict scientific conditions and reported on in the literature. Although in theory many of these methods may provide improvement, it is impossible to actually determine without proper empirical evidence. Therefore the final proposition from this review of requirements elicitation theory is that there currently exists a real and continuing need for the development and evaluation of approaches and tools for requirements elicitation with process guidance, that reduce the complexities, improve the results, and are both useable and useful to practitioners for the majority of software projects. As a result, the following chapter surveys requirements elicitation in practice, from the perspective of the available literature, as well as expert and novice analysts.

CHAPTER 3: A Survey of Practice

3.1 Chapter Overview

In the previous chapter we presented a thorough review of the relevant literature on and around the topic of requirements elicitation. We defined requirements elicitation with respect to software development projects, and described the fundamental activities of the requirements elicitation process. We then performed a critical analysis of the state of the art in requirements elicitation, focusing on the available techniques, approaches, and tools.

In this chapter we will support, enhance, and further refine the findings from our review of theory by first investigating what has been stated about requirements elicitation practice in the available literature (Section 3.2). We will then present a qualitative and quantitative survey of requirements elicitation practice consisting of a series of in-depth interviews with experts in requirements elicitation (Section 3.3), followed by an online questionnaire aimed at novice analysts (Section 3.4). This will be followed by a summary of the entire chapter (Section 3.5).

The purpose of this chapter is to investigate the state of practice in requirements elicitation (**Research Goal 2**), and what is generally perceived but not proven about requirements elicitation in practice (*Research Question 4*). This survey also acts as an elicitation session to increase our understanding of requirements elicitation in practice from the perspectives of both expert (*Research Question 5*) and novice (*Research Question 6*) analysts.

3.2 Practice in the Literature

3.2.1 Section Overview

In this section we follow on from our review of requirements elicitation theory in the previous chapter, to investigate requirements elicitation in practice through the available literature on and around this subject. We examine why requirements elicitation is so critical and complex, and look at the different roles the analyst must play during this important and difficult activity. We then uncover the core trends in requirements elicitation practice today, followed by a summary of the key issues and challenges often encountered. The purpose of this section is to identify from the literature some of the more commonly held perceptions about requirements elicitation in practice, and provide a foundation for the subsequent and more detailed survey of practice consisting of expert interviews and a novice questionnaire.

3.2.2 Why is requirements elicitation so hard?

More than half of the projects for software systems fail in terms of either being a) cancelled before completion or never implemented, or b) are completed and operational but over budget, over time, or with fewer features (The Standish Group 1994). In addition, more of these projects fail as a result of problems with requirements than for any other reason (Rolland & Prakash 2000; The Standish Group 2003). Although there is some debate over the exact figures, there is no doubt that errors made during the requirements phase of software projects are significantly more expensive to correct the further down the development process they are discovered (Boehm, B. 1981), and requirements errors are probably the most costly of any other kind. Van Lamsweerde (van Lamsweerde 2000) provides substantial evidence as to the difficulty of requirements engineering and its criticality in terms of its “utmost importance” with respect to software development.

Furthermore, it is generally accepted that the quality and success of a software system depends on the quality of the requirements upon which it has been built (Lloyd,

Rosson & Arthur 2002), and how well the final system meets those requirements (Moore & Shipman 2000). Therefore if the requirements do not satisfy the problems the system is intended to address, then the chance for project success is very small. In fact poor execution of elicitation will almost guarantee that the final project is likely to be a failure (Hickey & Davis 2003c). Poor requirements will reduce the quality of the software, introduce defects, create costly rework, cause late delivery of the system, and create customer and user dissatisfaction. High quality requirements on the other hand might not guarantee the results from the development of a system, but they do play a large role in its ultimate success (Gambhir 2001). They enable costs to be more accurately estimated, schedules to be more accurately estimated, and provide the basis of the design, and direction for project management. Subsequently due to the "central role" of requirements in software development, increasing their quality has the largest potential impact on the success of a system (Davis, A. M. 1993).

More specifically, the elicitation of the requirements is a complex and difficult process that is also critical to the overall success of most software development processes (Chatzoglou 1997; Cucchiarelli, Panti & Valenti 1994; Moore & Shipman 2000). Since elicitation often proceeds all of the other Software Engineering phases, its effectiveness and efficiency is of "pivotal importance" to the entire development lifecycle (Cybulski 1999). Furthermore, bad requirements as the result of poor elicitation make the rest of the software development process more problematic and increase the overall risk of failure. Good requirements elicitation on the other hand can decrease the amount of rework, improve productivity, enable better control over scope creep and requirements changes, increase customer satisfaction, and reduced maintenance and support costs (Wiegers 2003).

Although seen as a fundamental part of the process, requirements elicitation is often considered a major problem area and one of the biggest bottlenecks in projects for the development of software systems. During this process the analyst is required to manage large amounts of mostly raw information from a variety of sources, of many different types, some of which will inevitably be poor quality in terms of being incorrect, incomplete, and inconsistent. Its success depends heavily on the analyst addressing many technical, social, and contextual factors (Chatzoglou 1997), as well as the commitment and cooperation of the stakeholders. Consequently, the conditions

under which requirements elicitation is performed are never exactly the same twice. It can therefore be seen that this process requires an extensive skill set combined with experience to be performed well.

3.2.3 Roles of the analyst

As previously stated, the quality and success of requirements elicitation depends heavily on the experience and expertise of the participating analyst. Nuseibeh and Easterbrook state that “the tools and techniques used in RE draw upon a variety of disciplines, and the requirements engineer may be expected to master skills from a number of different disciplines”(Nuseibeh & Easterbrook 2000). They go on further to say that “the requirements engineer must possess both the social skills to interact with a variety of stakeholders, including potential non-technical customers, and the technical skills to interact with systems designers and developers”. Primarily because the elicitation of requirements is such a human-centred activity, analysts need to have excellent interpersonal, communication, analytical, and organisational skills. In this subsection we examine the various roles that analysts may be required to play when performing requirements elicitation for software systems. It is important to note that the analysts may not necessarily carry out all of these roles within all projects. The responsibilities of the analyst are dependent on the project and the context in which it is conducted.

3.2.3.1 Manager

A fundamental part of RE is related to project management. Analysts must monitor and manage the process of requirements elicitation, and communicate its progress effectively to the system stakeholders. This activity involves more than the obvious decision-making and prioritization tasks. Analysts are often required to initiate meetings with stakeholders, produce agendas and status reports, and remind stakeholders of their responsibilities. In many cases the analyst is the primary contact for questions from stakeholders relating to the project, the process, and the target system, and is responsible for communicating the goals of the system from the project sponsors to the other stakeholders.

3.2.3.2 Analyst

A large part of elicitation involves analysing not just the processes that the target system must support, but the requirements themselves. Analysts must translate and interpret the needs of stakeholders in order to make them understandable to the other stakeholders. Requirements are then organized in relation to each other, and given meaning with respect to the target system. Often the analyst is required to use a certain amount of introspection when eliciting requirements, especially when stakeholders are not able to express their needs clearly, or are unfamiliar with the available solutions. This typically involves modelling various aspects of the system and stakeholder knowledge at least mentally in order to investigate the requirements further and improve their quality. The analyst must therefore be able to uncover not only what users say they want, but what they really need (Davis, A. M. 1990).

3.2.3.3 Facilitator

Requirements engineers often need to play the important role of facilitator. When eliciting requirements by group work sessions, they are not only required to ask questions and record the answers, but must guide and assist the participants objectively in addressing all the relevant issues in order to obtain correct and complete requirements information. They are also responsible for ensuring that participants feel comfortable and confident with the process, and are given sufficient opportunity to contribute. The analyst should promote enthusiasm in the group and ensure that all stakeholders are committed and cooperative. The analyst must also act as a coach to the stakeholders, providing moral support and motivation to participate. Robertson (Robertson, J. 2002) argues that analysts should not be passive participants at all, but actively involved in the “invention” process of requirements elicitation, thereby creating ideas themselves and acting as a muse to encourage creativity from the other stakeholders. As a result this role represents a significant part of the skill required by analysts in order to perform effective requirements elicitation (Macaulay, L. A. 1999).

3.2.3.4 Mediator

During elicitation, conflicts between requirements as well as among stakeholders are inevitable. In many cases the prioritization of requirements from different stakeholders groups is a source of much debate and dispute. When these situations occur the analyst is often required to play the role of a mediator, and is responsible for finding a suitable resolution to the conflict through negotiation and compromise (Macaulay, L. A. 1999). It is important that the analyst is sensitive to all the social, political, and organisational aspects of the project when mediating discussions related to the target system. This involves resolving or at least minimizing power struggles and assertions of influence over the process by certain stakeholders. With respect to the actual requirements of the system, the analyst should encourage stakeholders to express their needs in terms that can be understood, validated, and verified by the other stakeholders.

3.2.3.5 Developer

Analysts are often required to assume the various roles of the developer community during requirements elicitation including the roles of system architects, designers, programmers, testers, quality assurance personnel, implementation consultants, and maintenance administrators. This is often due to the fact that these stakeholders have not yet been assigned to the project at the requirements elicitation stage. Despite this, the decisions made during this phase of the project will significantly affect these stakeholders, and the subsequent phases of development. Therefore it is important that they are represented in this process either directly or by proxy through the analyst.

3.2.3.6 Documenter

Frequently requirements engineers are responsible for documenting the elicited requirements, typically as a requirements specification or a detailed description of the target system. This role is particularly important as it represents the production of results and the output from the elicitation process, and forms the foundation for the subsequent project phases. Evaluation of the elicitation process and the work

performed by the analyst is based on the artefacts produced, which in some cases may form the basis of contractual agreements.

3.2.3.7 Validator

All the elicited requirements must be validated against the other stakeholders, existing systems, and each other, then compared with previously established goals for the project. By this it is meant that the requirements describe the desired features of the system appropriately, and that those requirements will provide the necessary functions in order to fulfil the specified objectives of the target system. This process typically involves all the identified stakeholder groups, and results in further elicitation activities.

3.2.4 Current trends in practice

Apart from the many different roles the analyst must play, the degree of difficulty in software development and requirements elicitation projects depends on a variety of factors including the number of functions, the amount of data, the interactions with other systems, and the number of stakeholder groups (Pfleeger 1991). Furthermore, analysts are being asked to produce better quality results for more complex systems with less time and resources. With respect to requirements elicitation in practice, the major issue appears to be that in most cases it is simply not done to a sufficient extent, and relatively little effort is devoted to this area. There are a number of possible reasons why requirements elicitation is still not being performed well in practice. Hickey and Davis (Hickey & Davis 2002, 2003a) have suggested that 1) the available methods are too complex and are not sufficiently useful, 2) analysts do not know alternative methods exist, and do not know how or when to apply them, and 3) analysts are content doing what they are doing and not interested in new methods. Regardless of the cause, the resultant effect is generally poor quality requirements and unsuccessful systems.

As previously implied, RE and more specifically the process of requirements elicitation, is not universally practiced as a distinct phase within the software

development lifecycle. However over the past decade or so, many of the more technically mature organisations have discovered that it is in their best interests and those of their customers, to invest the required time and effort into this phase by implementing a sufficient degree of structure and rigor to the process (REDEST 2003). Despite this, most project managers still make the fatal mistake of believing systematic requirements elicitation is a luxury rather than a necessity, especially when under increased schedule and budgetary pressures. As a result, requirements elicitation is often performed in an ad-hoc basis with no defined process (Neill & Laplante 2003). Furthermore, for many organisations capturing and defining requirements is primarily the responsibility of non-technical personnel or sales and marketing departments. This situation is inherently problematic as relevant technical knowledge is essential when formulating requirements that may impact the basic functionality or architecture of the system.

With respect to the selection of techniques used during a project, in practice it is more often determined by the experience and expertise of the analyst, rather than their appropriateness to the specific situation. The majority of analysts assigned the responsibility of eliciting requirements for software systems still use more traditional techniques, and in particular interviews and group workshops. One of the conclusions from the research of Hickey and Davis (Hickey & Davis 2003b) into requirements elicitation technique selection, with nine true experts of requirements elicitation and systems analysis in practice, was that “in general it appears that collaborative sessions are seen by most to be a standard or default approach to eliciting requirements”. Recently however the environment in which the system is to be situated is being given greater consideration through the use of contextual techniques. This is supported by Andreou (Andreou 2003) who points out that among expert analysts at least “the fundamental rules for collecting requirements are shifting in importance from the data to be processed by the system and the operations which process that data, to human-computer interaction and social and organizational factors”.

Of the available approaches specifically developed for requirements elicitation, JAD, Use Cases, Goals, and Scenarios have continued to grow in popularity and usage in practice over the past decade, at least among experienced practitioners. The reason for this is that despite their relative complexity, in general they have shown to be

effective in overcoming many of the issues often associated with requirements elicitation. Collaborative approaches in particular have also been found to be very helpful and successful in not only producing quality requirements, but also in achieving stakeholder buy-in and instilling project ownership (Gottesdiener 2002). We can see that these approaches, which have gained some degree of industry acceptance, consider the end users of the system during requirements elicitation more than traditional Software Engineering, where in the past it was primarily the developers and the customer who determined the requirements.

Although requirements management tools, which are largely administrative, have continued to receive acceptance in practice, the same cannot be said of tools for requirements elicitation. For the most part tools have only been used during requirements elicitation to facilitate group communication, or to support a specific technique. Although elicitation tools do exist to some extent, as we have seen in the previous chapter, in general practitioners have found them too complex to use, and do not appreciate the value, if any, they can provide. With only minimal perceived advantage, and a general lack of exposure to industry, the adoption and use of these types of tools in practice is very rare (Kasirun 2005).

In a study on the factors affecting the successful completion of the requirements elicitation stage in software development projects in practice (Chatzoglou 1997), the conclusions were that not enough resources are involved or allocated, and that more resources need to be allocated to the first iteration in an attempt to reduce the final number of iterations in the project. Furthermore, it was determined that the quality of the available tools and techniques are not adequate, and the management style and techniques adopted initially in the requirements elicitation process do not always seem to be the most appropriate. Arguably the most concerning finding of this industry based study was that in many cases an approach to requirements elicitation was chosen not because of its characteristics or advantages, but simply because it was more convenient or profitable for the analyst.

3.2.5 Common issues and challenges

The practical issues involved in the process of requirements elicitation are well documented in the literature, and can be categorized in any number of different ways. In the work by Christel and Kang (Christel & Kang 1992), requirements elicitation issues are grouped into 3 categories being 1) problems of scope, 2) problems of understanding, and 3) problems of volatility. For the sake of explanation, in this subsection we have categorized some of the more commonly occurring issues and challenges in requirements elicitation faced by practitioners according to the aspect of requirements elicitation that they most relate to. These have been collected from a variety of sources in the literature including (Christel & Kang 1992), (Wiegers 2003), and (Gottesdiener 2002), as well as from the experience and observations of the researcher's previous work in the IT industry.

3.2.5.1 Processes and Projects

Each project is unique and no two requirements elicitation situations are ever exactly the same. The process can be performed as part of a custom software development project, COTS selection activity, product line definition, or existing system maintenance operation. Projects can range all the way from simple bespoke web-based applications, to large and complex market-driven enterprise information systems. The environment in which the process takes place can also vary greatly depending on the geographic distribution of stakeholders, and the familiarity of users with software systems. Furthermore the process of requirements elicitation is inherently imprecise. This is as a result of the multiple variable factors, vast array of options and decision, and its communication and socially rich nature. Arguably one of the most common project based requirements elicitation issue is that the initial scope of the project has not been sufficiently defined, and as such is open to interpretations and assumptions. Projects like all functions of a business are subject to change and influence from internal or external factors including economic, political, social, organisational, legal, financial, psychological, historical and geographical. Another problem is that the success of a requirements elicitation process can only really be

determined once the entire project has been completed, and the system has been in use for some time.

3.2.5.2 Communication and Understanding

Natural languages such as English, which are the normal form of communication during requirements elicitation, are inherently ambiguous and imprecise. Furthermore it is common that stakeholders have difficulty articulating and expressing their needs. In some cases this may be as a result of the analyst and stakeholders not sharing a common understanding of concepts and terms, or the analyst being unfamiliar with the application domain, problems, and processes. The ‘say do’ problem (Goguen, J. A. & Linde 1993), when the users are able to perform a task but not describe it, is particularly relevant to the process of requirements elicitation. Alternatively, because of political or social reasons, stakeholders may not want to say what they want, or do. Often stakeholders will have difficulty seeing new ways of doing things, and do not know the consequences or feasibility of their requirements. Stakeholders may understand the problem domain very well, but be unfamiliar with the available solutions, and the different ways in which their needs could be met. Furthermore, stakeholders sometimes suggest solutions rather than requirements. Operations that are trivial or constantly repeated by stakeholders are often assumed and overlooked, despite the fact they may not be apparent to the analyst and other stakeholders. Likewise, concepts that are clearly defined and understood by one group of stakeholders may be entirely opaque to members of another.

3.2.5.3 Stakeholders and Sources

Conflicts between the needs of different stakeholders are common and almost inevitable. Furthermore, stakeholders may not want to compromise or prioritize their requirements when these conflicts occur. Sometimes stakeholders do not actually know what they want or what their real needs are, and are therefore limited in their ability to support the investigation of possible solutions. Likewise, stakeholders can be adverse to the change a new system may introduce, and therefore have varying levels of commitment and cooperation towards the project. Often stakeholders do not

understand or appreciate the needs of the other stakeholders, and might only be concerned with those factors that affect them directly. Most stakeholders will only see part of the problem, and all the required knowledge may be spread across many sources, including stakeholders from different backgrounds, in a variety of forms and notations. It is also possible that not all the stakeholders will be available to participate in the requirements elicitation process. Like all humans, stakeholders can change their minds and perceptions independently, or as a result of the elicitation process itself. Because requirements elicitation is largely a social activity, it can be significantly affected by the personalities, opinions, attitudes, and views of the many and varied participating stakeholders.

3.2.5.4 Experts versus Novices

One of the major problems in requirements elicitation is the significant gap between expert and novice analysts, and the subsequent lack of real experience and expertise among most practitioners in industry. This can be attributed to a number of factors, not least of which is the extensive skill set and range of experience necessary to perform this activity successfully. Novice analysts are typically those with limited elicitation expertise (i.e. have knowledge of only a few techniques and approaches), some degree of domain knowledge, but a narrow scope of practical experience (e.g. have participated in only a small number of projects). This may be as a result of a lack of education in terms of the theory behind the available techniques and approaches, or the lack of practice in using soft skills such as listening, communicating, and questioning. Experts tend to be those with the many years of experience, having applied the theoretical aspects of the discipline to many and varied practical situations, with knowledge of a range of methods, their relative strengths and weaknesses, and when and how to use them. Subsequently, true experts in this complex and critical activity are few and far between. In fact “most practicing analysts are more journeyman than master”, and it is therefore not surprising that more than half of the software systems developed fail to satisfy the needs of the users (Hickey & Davis 2003b).

3.2.5.5 Quality of Results

The requirements elicited from stakeholders may not be feasible, cost-effective, or easy to validate. Because the information is typically unstructured during elicitation, in many cases the requirements can be vague, lacking specifics, and not represented in such a way as can be measured or tested. Furthermore, requirements may be defined at different levels of abstraction and insufficient levels of detail. Because the process of elicitation is informal by nature, a set of requirements may be incorrect, incomplete, inconsistent, and not clear to all of the participating stakeholders. This may be because some requirements are unknown at the stage of elicitation, while others may be assumed, and therefore not mentioned by the stakeholders, or recorded by the analyst. Lack of a common understanding and informal communications can lead to ambiguity of the elicited requirements. In addition, the context in which requirements are elicited, and the process itself is inherently volatile. As the project develops and stakeholders become more familiar with the problem and solution domains, the goals of the system and the wants of the users are susceptible to change. In this way the process of elicitation can actually cause requirements volatility, and therefore affect the quality of the requirements as a whole.

3.2.5.6 Research versus Practice

The large gap between requirements elicitation research and practice can be attributed to a number of factors. These include the relative youth of the field, the directions of current research, and the ratio of practitioners to researchers. However a general lack of awareness by many analysts of the available methods for requirements elicitation, combined with a general unwillingness to adopt them, is largely responsible for this situation (Maiden, N. A. M. & Rugg 1996). Chatzoglou (Chatzoglou 1997) states that the difference between the theory and practice of requirements elicitation exists because “1) People who work for industry are not sufficiently informed about new methodologies and the way they should be used, 2) Methodologies are difficult to use in practice, mainly because they are not very well documented or an extensive training program is required before they can be put into use, and 3) Projects developed by industry are usually small-size projects which are developed for internal

use. People who develop them have the impression that they do no need any methodology".

3.2.6 Section Summary

RE and requirements elicitation has continued to attract more and more attention in practice, especially over the past ten years, as the emphasis has changed from 'building the system right' to 'building the right system'. The importance practitioners are placing on requirements elicitation during system development in general has increased, as has the availability of information on the state of the art requirements elicitation theory. From the range of existing techniques, variations of interviews, goals, scenarios, and especially group workshops, are still the most widely used and successful in practice. Regardless of how it is performed in practice, it is now generally accepted that effective and efficient requirements elicitation leads to the development of successful products and satisfied customers. Despite this, there still appears to be a deficiency of sufficient awareness, understanding, and expertise in requirements elicitation practice today.

Large gaps still exist between requirements elicitation theory and practice, as well as expert and novice analysts. Current methods for requirements elicitation are either unknown or too complex for most practitioners to use, and most have only limited implementation support. Practitioners have for the most part rejected those process, methods, and tools that are perceived as detailed or heavyweight. Furthermore, the process of requirements elicitation is rarely given the necessary amount of attention, budget, and schedule, required to perform it properly. Unfortunately, many in practice continue to make the same mistakes time and time again with respect to requirements elicitation, and do not acknowledge the real issues and their subsequent effects. The likely results of these situations are well documented in the literature and known in practice, and include costly rework, schedule overruns, project failure, poor quality systems, and customer dissatisfaction.

Regrettably very few practical solutions to requirements elicitation seem to have been successfully developed (Kasser 2004). Because requirements elicitation is one of the

most poorly executed software development activities in practice, there is a real need for rigorous and systematic approaches to provide the required level of effective support (van Lamsweerde 2000). Along these lines, Shaw and Gaines (Shaw & Gaines 1996) suggest that what is really needed are team-based approaches and support tools. A survey of requirements elicitation conducted in practice proposed that not only was there a lack of tools and approaches simple enough to be introduced in small and immature companies, but that such approaches and tools would increase requirements completeness and understandability (Karlsson et al. 2002). This highlights the importance of working towards reducing the gap between research and practice in terms of awareness, acceptance, and adoption, through practical approaches and guidelines that can be easily taught and used by novice analysts in particular. This can be achieved through the development of new and improved approaches, which reduce the complexity of the process, and offer appropriate tool support that requires less expertise.

We can see that relatively little attention has been devoted to case studies, experience reports, and industry stories specifically on the topic of requirements elicitation. In fact it has been noted that there is not much solid empirical work at all, and what there is, is too narrow to form the basis of a tool or method (Finkelstein 1994). Although there has been a number of retrospectives and state of the art summaries on RE in recent times such as (Nuseibeh & Easterbrook 2000) and (van Lamsweerde 2000), few if any have been concerned with requirements elicitation in particular, and its overall state of practice. Although Hickey and Davis (Hickey & Davis 2003b) investigated how experts select which elicitation technique to use and when, this work concerned only one critical decision in the process of requirements elicitation, namely technique selection. Despite this the contribution of the work to the field in better understanding the process of requirements elicitation was significant. Likewise the work of Christel and Kang (Christel & Kang 1992) previously mentioned, which provided invaluable information for both researchers and practitioners by looking in detail at issues commonly faced during elicitation, looked at only one aspect of the process. We therefore extend our survey of requirements elicitation in practice by conducting both expert interviews (Section 3.3) and a novice questionnaire (Section 3.4) in order to fill the gaps in the existing research, and directly address why

requirements elicitation in practice continues to be performed poorly, and what can be done to improve this situation.

In summary, the key points identified in this section are:

1. The elicitation of requirements is a very difficult and important activity, which is complex in terms of the number of factors that must be taken into consideration, and critical to the entire process of software systems development.

2. Although the core concepts of requirements elicitation continue to become more well known and widely practiced in industry, the process itself is still often conducted poorly and in an ad-hoc fashion, without the required level of attention.

3. Large gaps exist between both theory and practice, and expert and novice analysts, highlighted by the fact that most analysts still use traditional techniques, and the adoption of requirements elicitation specific approaches and tools is confined to a very small practitioner population.

4. Not only is an extensive skill set required for the analyst to perform all the different roles necessary during the process of requirements elicitation, but a wide variety of issues and challenges must also be handled in order to produce high quality results.

3.3 In-depth Interviews with Experts

3.3.1 Section Overview

In this section we present the process used and the results obtained from seven questionnaire-based interviews with experts in the field of requirements elicitation in practice, on the subjects of novice analysts, process guidelines, tool support, and approach evaluation. We examine the need for the development of new approaches and tools in terms of their composition, and how these may assist novice analysts during the early stages of requirements elicitation to improve the overall quality of the process and requirements produced. The purpose of this section is to investigate what requirements elicitation experts think of the current state of practice, and how it can be improved. We therefore aim to prove or disprove some of the common perceptions about requirements elicitation practice as reported in the literature, and also to elicit high-level requirements for the development of a new and improved approach and supporting tool.

3.3.2 Method

It was determined that interviews were the most appropriate research technique to use for the acquisition of knowledge, based primarily on the experience of experts (Agarwal & Tanniru 1990). Interviews enable the researcher to question the experts directly about their thoughts and opinions, and allow the experts the freedom to describe and reflect in detail on their views and beliefs. Furthermore, structured interviews, i.e. questionnaire-based, were deemed the most suitable because of the geographical distribution between the researcher and participants, and the heavy burdens on the time of the experts. This also provided the best solution in terms of the effort required for data collection and analysis (Kvale 1996).

An alternative to the use of interviews was ethnography (e.g. observation), however it was deemed unnecessary to invest the additional time and risk associated with this method. This is because we were primarily interested in only the core characteristics

and the general state of requirements elicitation practice as a whole, as opposed to a detailed analysis of specific factors for a small selection of project instances.

The subsections below describe the following 6-step research methodology used to design and conduct the expert interviews:

1. Determine specific research goals
2. Establish participant criteria
3. Develop the questionnaire
4. Pilot study and refinement
5. Contact potential participants
6. Data collection and analysis

3.3.2.1 Determine specific research goals

The first step was to determine the specific research goals for the expert interviews. The primary goal of the research was to investigate what type of approach and tool might be developed to support and improve requirements elicitation in the specified context. The interviews were specifically intended to explore not only what experts thought of the state of practice in requirements elicitation (*Research Question 5*), but also what they believed would be the key components of a new and improved approach and tool for requirements elicitation (*Research Question 7*). Furthermore it was anticipated that the results would in turn provide confirmation of the literature, and reaffirm or refute common perceptions of requirements elicitation practice.

3.3.2.2 Establish participant criteria

The next step was to establish a criterion for the participants, based on reasonable assumptions of what would constitute an expert, and support the type of knowledge sought after. As a result the criteria was very strict with participants requiring at least ten years experience in software development, with time spent in both academia and industry. They needed to have worked on a variety of projects in terms of the size and the types of systems being developed. Furthermore the participants needed to have

recently published internationally, specifically on requirements elicitation either in the form of a conference or journal paper, or as an author of a textbook that covered requirements elicitation. All participants had to be actively involved in the RE community, and be currently performing, teaching, or training RE including elicitation.

Although having such a demanding criteria significantly narrowed down the number of people suitable for the survey, it ensured that the qualifications and experience of participants as experts could not be challenged. In fact it could be said that anyone matching the criteria would surely be important and influential to the field of RE. Additionally, the criteria reduced the number of participants required for the results from the research to be valid, and still offer a contribution to the existing literature and future research.

3.3.2.3 Develop the questionnaire

By breaking down the specific research goals determined in Step 1, the individual questions were designed, with the total questionnaire being made up of 21 open-ended questions (see Appendix C), divided into 6 sections for each of the main topics covered as shown in Table 3.3.1 below.

Table 3.3.1: Questionnaire Overview

Section	***Title***	***Questions***
1	General Information	4
2	Experts and Novices	3
3	Process Guidelines	3
4	Tool Support	3
5	Approach Evaluation	3
6	Feedback	5
	Total:	21

Open-ended questions were selected as the most appropriate type because of the high-level nature of topics covered, and the possible variety and range of expert

experiences (Patton 2001). They also enable participants more flexibility with respect to how they wished to answer the questions. In many cases open-ended questions were the only real possibility in order to cover the chosen topics given the large number of known and unknown possible responses to each of the questions.

The number of questions was determined by a desire to keep the estimated time required to complete the interview to approximately ninety minutes. This was deemed to be an appropriate length of time given the necessary compromise between covering the desired topics in enough detail, and the amount of time the participants were able to give. The wording of the questions was also carefully developed in accordance with ethical considerations to ensure the participants could not be identified, and keep the responses anonymous. As with all interviews, the wording of the questions is paramount, and all possible measures were taken to be as concise as possible and avoid ambiguity so to not confuse the respondent, which included the pilot study described in the next subsection

3.3.2.4 Pilot study and refinement

A preliminary version of the questionnaire was piloted internally to determine any overlaps in the questions, relevancy to the desired topics, as well as consistency in phrasing, and clarity of explanation. This was achieved by conducting a face-to-face interview using the questionnaire, with a participant who was external to the research, but who matched the established expert criteria. The pilot study participant had the added advantage of being someone who has significant experience in interviewing requirements analysts and being interviewed as a requirements analyst. Based on the results from this pilot study and the feedback from the external participant, the questionnaire was refined and formally documented.

3.3.2.5 Contact potential participants

A list of sixteen people known to the researcher and principal supervisor that matched the criteria was developed. All sixteen were contacted via an Invitation to Participate email (see Appendix A), with twelve replies being received, and all agreeing to

participate. The twelve respondents to the Invitation were once again sent via email a consent form (see Appendix B) to be returned electronically in accordance with the university's ethics requirements, and an Introduction email with the questionnaire attached.

The Information email gave respondents the option of completing the questionnaire by way of interview (face-to-face or telephone), or offline electronically. All twelve participants selected to complete the questionnaire offline and return via email due to a number of reasons including geographical and time differences, and the ability it offered to reflect on the questions and provide carefully thought out answers. The offline option made it easier for the task of completing the questionnaire to fit into the busy and broken schedules of the experts.

3.3.2.6 Data collection and analysis

From the twelve experts that agreed to participate out of the sixteen contacted via email, only seven actually completed and returned the questionnaire within the specified time period for the data collection stage of this phase of the research. Apart from the emails containing the completed questionnaires, only two experts asked for further clarifications on some of the questions, but six offered to be contacted if further clarification of their answers was desired or required.

All the completed questionnaires were given a numerical identification code, for the purposes of confidentiality, based on the sequence they were received. The seven responses were then collated into a single document, with the individual answers grouped together by question. Each question was then analysed individually using content analysis (Krippendorf 2004) to record the number of times particular terms were used, and to identify common and important themes in the responses. Content analysis was selected because it enabled us to identify trends and patterns in the responses of the experts both through the frequency of key words, and via groups of words with similar meaning or connotations (Stemler 2001).

The use of a qualitative software tool to support the data analysis of the expert interviews was considered, and several options were examined including Nvivo (QSR International 2006). However the decision was taken to use a manual approach to the coding of themes and terms required during content analysis of the expert interview transcripts, due to the relatively small volume of data collected, and the additional effort required to perform the data analysis using a specialised qualitative software tool.

3.3.3 Results

For the sake of consistency in the analysis and presentation, the subsections below each represent a section in the questionnaire (as shown in Table 3.3.1 and Appendix C), and contain a summary of the questions asked, followed by an analysis of the responses.

3.3.3.1 General Information

In Section 1 (General Information) of the questionnaire, respondents were asked about their experience and expertise with respect to requirements elicitation, their comments on the current state of practice including major trends and challenges, and what types of new approaches if any they believed were needed and why.

In response to the question about their **experience and expertise** in requirements elicitation, the total number of years in the software industry from the seven respondents was 197, at an average of 28.14 per respondent. The average number of projects the respondents had each been involved with was 55, with a total of 385 for the group. From the seven respondents, two were from the USA, two were from Australia, and the remaining three respondents were European.

Projects ranged in length from 10 days to several years, and in value from US$5K to several million. The types of projects and systems the respondents had been involved in included the development and implementation of information systems, control systems, simulators, operating systems, compilers, embedded software, and industrial

solutions. The types of industries these projects were performed in were just as wide ranging, and included retail, telecommunications, manufacturing, medical, and government. The various roles performed by the respondents in these projects were also varied, and ranged from advisor, analyst, consultant, and manager type roles, to reviewer, designer, programmer, testers, and quality assurance.

When asked about what **general comments** the respondents would make about the current state of practice in requirements elicitation from their personal experience, all seven identified the need for improvement. Suggestions for how this could be achieved included the adoption of more systematic processes and guidelines, improving the education and expertise of analysts, and improving the application of techniques and usage of tools. Despite this, four of the respondents identified that the RE research community had already made notable progress leading to improvements in practice.

With respect to what the respondents saw as the **major trends** in requirements elicitation practice today, the responses were quite varied. Some referred to specific approaches such as iterative, Agile, and Object-oriented development methodologies, and the increasing reliance on techniques such as Use Cases. Others expressed more general views such as the attempts to formalize processes, and the opinion that systems in general were getting more complex and expensive to build.

Likewise, with respect to the **major challenges**, the responses were also diverse and included risk identification, early discovery of major requirements, handling integration requirements to third party systems, and increasing the adoption of new techniques and approaches into mainstream practice. One respondent mentioned that "*analysts not really listening to the customers is both a major trend and challenge*". Similarly, another respondent stated that "*having better educated analysts was a trend, but producing well-educated analysts was still a challenge*".

Only three of the seven respondents believed that there was a need for **new approaches** to requirements elicitation in practice. Suggestions offered for the major elements of these approaches included "*the implementation of more systematic guidelines*", and "*the ability to employ the approach at different levels of*

abstraction". Another idea for a new approach included the ability to zoom between levels of detail, with the ability to *"view the synopsis all the way through to the script, to use a film analogy"*. Other proposals put forward were the ability to manage the communication between the different stakeholder groups, and facilitate their mutual understanding of the requirements. It was recommended by one respondent that any new approach should be flexible to the process, but specific to the context. Tool support with appropriate training was also considered important for new approaches.

The remaining four respondents stated that the existing approaches in theory were sufficient, however their application and usage in practice needed to be improved. Suggestions for how to make this happen included involving more stakeholders, and really understanding their needs, as well as improving the way in which analysts are educated and trained on how to select and use the existing approaches. One respondent suggested that what was really needed, with respect to new approaches, was not the production of specifications that are larger with more detail, but shorter and closer to the business goals and user tasks.

3.3.3.2 Experts and Novices

In Section 2 (Experts and Novices) of the questionnaire, respondents were asked about their definitions of expert and novice requirements elicitation analysts, the major differences between them, what they believed were the most common mistakes made by novices, and what types of support were needed to help novices become experts.

In response to the question on how would the respondents define an **expert analyst** of requirements elicitation, and what are the major differences between them and novices, five of the seven respondents referred in general to the amount of knowledge (e.g. number of techniques known and when to use each one) and experience (e.g. number of projects, types of projects, types of organisations, and different techniques used) as the key-determining factors between experts and novices. Conversely, a novice was seen as having only limited experience, and is competent in only a limited number of the available techniques.

Two of the respondents referred to successful results (i.e. customer satisfaction and return on investment), and an effective elicitation process, as ways of determining if an analyst is expert. One of these respondents qualified the answer by saying this definition may restrict the analysts expert qualifications to a specific domain or application area only. Two of the participants also referred to the way in which elicitation is performed as a determining factor for an expert versus a novice. They stated that *"an expert will be more thorough and check their understanding"*. Also that *"an expert will tend to know or be able to find those areas often overlooked"*, and *"have the ability to understand and successfully combine knowledge of the business needs with those of the users"*.

When asked about their opinion of what are the most **common pitfalls** and mistakes made by novice analysts, the responses were as follows:

- 3 respondents stated that listening to, but not really learning from, and subsequently understanding the stakeholders, business goals, users environment, and activities, were the most common novice mistakes
- 3 referred to novices solely relying on just a few core techniques for all situations and all times as a frequent pitfall
- 3 noted that not managing the stakeholder relationship effectively was another typical mistake made by novices, whether it meant not including them appropriately or effectively, or not explaining to them the value of the process, the project, and their involvement
- 1 respondent cited blaming the customer for their inability to articulate what they want as something novices do, and another suggested that novices often have a solution in mind during elicitation, and drive the process to make the requirements fit that solution

With respect to what **types of support** did the respondents believe are needed to help novices perform requirements elicitation better and eventually become experts, three of the respondents suggested that tools were a useful form of support for novices. Five of the seven respondents stated that more training and/or mentoring with experts was needed for them to become experts. Two referred to experience as being of critical

importance, and one respondent suggested that an effective way to encourage novices to become more expert was to use checklists and ongoing reviews to ensure that the analysts are applying the appropriate amount of effort to the process in accordance with the training and mentoring they had received.

3.3.3.3 Process Guidelines

In Section 3 (Process Guidelines) of the questionnaire, respondents were asked about the importance of developing a process for requirements elicitation, to what extent did they believe structure and rigor could be employed during the early stages, and how they would typically perform requirements elicitation in the described context.

All respondents believed that it is important to **develop a process** for the early stages of requirements elicitation, although three stated that such a process should be more like a set of guidelines and be lean and not fixed. The result of not having a defined process as described by the respondents included *"haphazard"*, *"flailing around"*, *"unsystematic"*, and *"discovery of the irrelevant"*. One respondent explained that because there are certain tasks that should be performed in every requirements elicitation project, developing a process was therefore important to ensure they were all completed. Likewise, another respondent suggested that developing such a process enabled best practices to be encouraged and permitted, thereby building on the current state of knowledge in practice.

In response to the question about the extent to which the respondents believed **structure and rigor** could or should be employed in the early stages of requirements elicitation, five said that a form of structure or rigor would need to be kept at a *"general"* or high level. Three respondents were of the opinion that structure and rigor had to be balanced with flexibility. Furthermore, two affirmed that it should be possible to *"tailor"* the process for each particular case. Three respondents stated that structure and rigor if pushed too far could actually stifle the elicitation process, with one saying it could even be a "*project killer*", and another suggesting it "*could almost become an obstacle for creativity"*.

When asked about how would the respondents typically **perform the process** of requirements elicitation in the context described, in terms of the activities performed and the techniques used, three listed the high level tasks they would perform (e.g. identify and describe stakeholders, and review current relevant operational processes). Two respondents referred to their own personal publications on the topic that provided similar types of information, namely a general roadmap with guidelines for various tasks to be performed during elicitation. Another two of the respondents declared that they would develop and adopt a process specific to the situation, and one would *"sense the situation and use my gut"*, while the other had *"no explicit rules"* and would *"use intuition a lot"*.

3.3.3.4 Tool Support

In Section 4 (Tool Support) of the questionnaire, respondents were asked about what types of tools they had used for requirements elicitation, how effective they had been, and what features and activities did they believe would be useful to support in a tool for analysts and stakeholders during requirements elicitation.

With respect to what **types of tools** had the respondents used for the elicitation of requirements and how effective have they been, two respondents stated that they did not use any tools for requirements elicitation, while a further three said they mostly only used generic tools such as word processors and spreadsheets. Three of the respondents had seen or heard of specific tools being used during requirements elicitation, but not really for the actual task elicitation of requirements itself, such as Requirements Management and CSCW applications. One respondent mentioned that he had been the development project leader for a tool that had been used successfully for elicitation, although active instruction was required for the analyst before it could be used effectively. Apart from this respondent, none of the remaining six cited specific examples in their experience where a requirements elicitation tool had been used successfully.

Flexibility was mentioned by three respondents as a **useful attribute** in a software tool to support the process of requirements elicitation in the context described, as was

ease of use. Two suggested that guidance on what to write in a requirements statement and specification template would also be useful. Other specific features offered by the respondents included communication and consensus support, the capacity to relate requirements to business goals, and the ability to capture and retrieve information about the product, users, activities, and environment. Two respondents stated that it was difficult for them to answer this question because they could not really think of any specific features.

In response to the question on what activities did the respondents believe an intelligent **software tool** could support the analyst and stakeholders perform during requirements elicitation, the responses were many and varied, and most had an implication of a suggestion rather than a recommendation. Specific examples were:

- Managing the level of abstraction or detail
- Managing stakeholder communication
- Organisation of ideas and information
- Identification of potential "holes"
- Generating relationships between requirements, scenarios, and business goals

However, three respondents expressed their difficulty in knowing how exactly some of these could be actually implemented. Two of the respondents were of the opinion that tool intelligence could only be used to offer advice on the process, provide guidance on things like technique selection, or educate the user on a particular technique, but could not support the actual elicitation of requirements itself.

3.3.3.5 Approach Evaluation

In Section 5 (Approach Evaluation) of the questionnaire, respondents were asked about how they would define and measure successful requirements elicitation, and what metrics would be appropriate to evaluate a new approach to requirements elicitation and compare it with existing approaches.

When asked how the respondents would define and **measure the success** of a requirements elicitation process, four out of the seven stated that *"customer and/or*

stakeholder satisfaction" was key, with one stating that they were of the opinion there was a strong correlation between stakeholder happiness during elicitation and project success. Four out of the seven respondents also stated that the quality of the requirements set with respect to their completeness, correctness, clarity, and the number of variations/changes requested/required would also be a valid way of measuring requirements elicitation success. It is important to note that both of these relate to the results of the entire project, well after the elicitation phase has been completed. One respondent made reference to the customers having realistic expectations of what they were going to get, and how it would effect their work and their goals, as a suitable measure for the success of the project, and at the same time the developers/suppliers understanding this and being able to meet those expectations at an acceptable cost. Another respondent mentioned that although it has been used sometimes in practice, the measure of *"the number of requirements agreed per hour"* was a particularly bad one to use, and not appropriate at all.

With respect to what ways did the respondents believe a new approach to requirements elicitation could be **evaluated and compared** to existing approaches, three respondents referred to their answer for the previous question, whereby measurements such as completeness, correctness, clarity, customer satisfaction with the product, and the number of changes could be used as ways of evaluating a new approach against others. Two respondents commented that a subjective review of the resultant project artefacts, such as documents, prototypes, and early versions of the software by customers and other stakeholders involved, could also be used to evaluate and compare other experiences. Another two of the respondents suggested that the approach could only be considered successful after practitioners conducted a number of *"real life"* projects in *"real time"*. One respondent suggested that experiments in an academic setting could be performed as a way of evaluation, where the new approach and existing ones are used in the same settings. Only three of the seven referred specifically to customer satisfaction as a metric they believed would be appropriate for the evaluation of a new approach to requirements elicitation. Other specific metrics suggested included the number of defects resulting from poor requirements, additional time spent on requirements elicitation after the elicitation process has been completed, and the number of changes to the requirements after this

point. Three respondents indicated that they were not sure of which specific metrics would be appropriate to use in this case.

3.3.3.6 Feedback

In Section 6 (Feedback) the respondents were ask to provide some feedback on the questionnaire mainly to determine if the interviews were deficient in any way. Six out of the seven respondents believed that the questionnaire was of an appropriate length, with completion times ranging from 45 minutes to 2 hours, and at least one respondent completing the questionnaire in several separate sessions. The remaining respondent stated that the questionnaire *"could have been longer without difficulty"*. Similarity, six out of the seven respondents believed that the questions were sufficiently detailed, with two respondents making the comment that this really depended on how the information from the responses was going to be analysed and used.

3.3.4 Discussion

The interviews presented above have shown that opinion on the current state of practice in requirements elicitation, even among experts, is generally not positive, and would imply that significant attention and effort is still needed in this area. This was not unexpected given the overall theme of surveys in the literature, and common perceptions in industry. The fact that a number of different major trends and challenges were mentioned may suggest that there is in reality a significant and varied amount of them in practice today. Opinion on the need for new approaches versus better implementation of the existing ones was split, although all respondents acknowledged the need for improvement and better support.

It is not surprising that a combination of significant experience and expertise is necessary to become expert in requirements elicitation. However, it is interesting to note that the ability to manage and understand stakeholders would appear to be a major differentiation between the experts and novices. It was widely agreed that

closing this gap involved more training and mentoring, which was consistent with the other responses from that section of the questionnaire.

As expected, it can clearly be identified that experts see developing a process for requirements elicitation as important. However, it is notable that the structure and rigor of the process should not restrict the analyst, or inhibit the creativity of the stakeholders. This reflects the typically generic way in which strict software development processes are applied to largely social problems. When called upon to perform requirements elicitation, some experts would follow a basic core process, where as others would more likely 'feel' for the appropriate way to proceed.

We can see that the use of specific requirements elicitation tools among experts is not widespread, and when respondents were pressed to list specific features that might be useful in such a tool, there was substantial difficultly in identifying or imagining them. Similarly, the experts struggled to envisage which requirements elicitation tasks would benefit from an intelligent tool at all. However, this is not unexpected given that most experts depend primarily on their experience, understanding, and interpersonal communication skills throughout the requirements elicitation process, and it is very often the level of these skills that differentiate them as expert analysts from novices. It may also be the case that experts cannot imagine how a tool could possibly replace them and their role during the process of eliciting requirements.

Limitations

As with all questionnaire-based research, the opportunity for follow-up questions, response clarification, and potentially useful further in-depth discussion are not possible. However, the quality of the participants and the frequency with which in normal activities they are required to document their thoughts and opinions, somewhat reduces this disadvantage. The number of respondents and the effectiveness of the questions could also be considered as restrictions on the research, but this has been minimized by the strict criteria for participation, and the iterative analysis and refinement of the questionnaire. Those respondents, whose research and practice activities have involved the development of tools and processes for requirements elicitation, will generally be of the opinion that this is not only a worthwhile endeavour, but is also needed. Cultural and educational biases may also

be a factor, as some respondents originate from more technical and process oriented environment, whereas others hail from a more sociological and design background. In addition, the memory and opinions of the respondents may be overly affected by their more recent experiences, or by their most significant successes and failures.

3.3.5 Section Summary

In summary, the key points identified in this section are:

1. The need for improvement in requirements elicitation practice was unanimous, however there was no consensus on how this can be achieved

2. The most common mistakes made by novices appear to be to those related to managing the stakeholders

3. More training and tools are required to support novice analysts during the process of requirements elicitation

4. All the experts believed developing a process of requirements elicitation is important, however the guidelines should be flexible and at a high-level

5. Tools are rarely used by both expert and novices, however the potential exists for them to support a variety of requirements elicitation activities

6. The success of a requirements elicitation process should be based on the quality of results and the satisfaction of customers

7. New requirements elicitation approaches should be validated through controlled experiments and real world projects

3.4 Online Questionnaire for Novices

3.4.1 Section Overview

In this section we present the process used and the results obtained from an online questionnaire completed by twenty-three novice practitioners in the field of requirements elicitation, on the subjects of technique and tool selection and usage, as well as support and assistance. We examine the need for the development of new approaches and tools, in terms of their composition, and how these may improve the performance of novice analysts, and the overall quality of the process and requirements produced. The purpose of this section is to investigate what requirements elicitation novices think about the current state of practice and how it can be improved. Much like the expert interviews described in the previous section, the aim of this section is therefore to prove or disprove some of the common perceptions about requirements elicitation practice, and also to elicit high-level requirements for the development of a new and improved approach and tool.

3.4.2 Method

It was determined that a questionnaire was the most appropriate research technique to use for the acquisition of knowledge, based primarily on the experience of novices (Foddy 1994). A questionnaire enables the researcher to question the novices directly about their thoughts and opinions, and allows the novices to easily and quickly record their views and beliefs. Furthermore, an online questionnaire, i.e. web-based, was deemed the most suitable because of the geographical distribution between the researcher and the participants, and the convenience of being able to complete it at any time from any place. This also provided the best solution in terms of the time and effort required for data collection and analysis.

Alternatives to the use of a questionnaire were ethnography (e.g. observation) and interviews, however because of the anticipated number of participants required, it was once again deemed unnecessary to invest the additional time and risk associated with

these methods. This is because we were primarily interested in only the core characteristics and the general state of practice in requirements elicitation as a whole, as opposed to a detailed analysis of specific factors for a small selection of project instances.

The subsections below describe the following 6-step research methodology used to design and conduct the novice questionnaire:

1. Determine specific research goals
2. Establish participant criteria
3. Develop the questionnaire
4. Pilot study and refinement
5. Contact potential participants
6. Data collection and analysis

3.4.2.1 Determine specific research goals

The first step was to determine the specific research goals for the novice questionnaire. The primary goal of the research was to investigate what type of approach and tool might be developed to support and improve requirements elicitation in the specified context. The questionnaire was specifically intended to explore not only what novices thought of the state of practice in requirements elicitation (*Research Question 6*), but also what they believed would be the key components of a new and improved approach and tool for requirements elicitation (*Research Question 7*). Furthermore it was anticipated that the results would in turn provide confirmation of the literature, and reaffirm or refute common perceptions of requirements elicitation practice.

3.4.2.2 Establish participant criteria

The next step was to establish a criterion for the participants, based on reasonable assumptions of what would constitute a novice, and support the type of knowledge sought after. As a result the criterion was very broad with participants requiring less

than five years industry experience in the Information Technology area. Furthermore, the participants need to have been directly involved in the process of capturing and collecting (eliciting) requirements for software development and/or implementation projects from customers and/or users. These criteria ensured that participants would not only be novices, but that the questionnaire would cover a large proportion of the analyst community, whilst producing valid results that would offer a contribution to the existing literature and future research.

3.4.2.3 Develop the questionnaire

By breaking down the specific research goals determined in Step 1, and taking into consideration the available literature and the results from the expert interviews, the individual questions were designed, with the total questionnaire being made up of 16 multiple-choice questions (see Appendix B), divided into 4 sections for each of the main topics covered as shown in Table 3.4.1 below.

Table 3.4.1: Questionnaire Overview

Section	***Title***	***Questions***
1	General Information	6
2	State of Practice	5
3	Techniques and Tools	3
4	Assistance and Support	2
	Total:	16

Multiple-choice questions were selected as the most appropriate type because they enable the participants to answer the questions easily by simply selecting from a list of available responses (Rea & Parker 2005). It was anticipated that this would make the questionnaire more attractive to novices, and therefore encourage more of them to participate in the research. Furthermore, multiple-choice questions were the only real possibility for effectively and efficiently covering the chosen topics at the desired high-level, and subsequently processing the resultant answers.

Wherever possible a Likert scale (Oppenheim 1992) was used for the available answers to the questions. The main motivation for this was the ability to easily answer (respondents) and analyse (researcher) the data once it has been collected. However, we acknowledge that there are limitations to using Likert scales in questionnaires (Page-Bucci 2003). Likert scales force respondents to make a decision or state an opinion even though they might not have one. Furthermore, a Likert scale heavily restricts the way in which the respondent can answer, and provides no indication if the respondent has interpreted the question correctly. Although there is some controversy about the effectiveness of using Likert scales in social research circles (Fowler, F. J. 1995), particularly when eliciting attitudes and opinions, because of the type of information we wanted to collect, and the level of detail required for the analysis, we believed that using a Likert scale (5 point in our case) was an effective and efficient approach.

The detail and number of questions was determined by a desire to keep the estimated time required to complete the questionnaire to approximately ten minutes. This was deemed to be an appropriate length of time given the necessary compromise between covering the desired topics in enough detail, and the amount of time the participants would be willing to give. The wording of the questions was also carefully developed in accordance with ethical considerations to ensure the participants could not be identified, and keep the responses anonymous. As with all questionnaires, the wording of the questions is paramount, and all possible measures were taken to be as concise as possible and avoid ambiguity so to not confuse the respondent, which included the pilot study described in the next subsection.

3.4.2.4 Pilot study and refinement

A preliminary version of the questionnaire was piloted internally to determine the coverage of available responses, relevancy to the desired topics, as well as consistency in phrasing and clarity of explanation. This was achieved by first getting a requirements elicitation expert to review the questionnaire in detail. We then asked three participants external to the research that matched the established novice criteria to complete the questionnaire and provide feedback. Based on the results from this

pilot study, and the feedback from both the expert and three novices, the questionnaire was refined and formally documented.

3.4.2.5 Contact potential participants

A total of forty-one people that matched the criteria and where either known by the researcher, or known of through people known to the researcher, were contacted via an Invitation to Participate email. This email contained the Internet link (URL) for the front page of the website containing the questionnaire implemented online using UCCASS (Holmes 2004), a freeware open source survey system. In accordance with the university's ethics requirements, no consent form was necessary since all responses were to be collected entirely electronically and anonymously.

3.4.2.6 Data collection and analysis

From the forty-one novices contacted via email, only twenty-three actually completed the online questionnaire within the specified time period for the data collection stage of this phase of the research. The twenty-three responses were then collated into a single document, with the individual answers grouped together by question. Each question was then analysed individually using frequency analysis to record the number of times each available response was selected, and the percentage of the total responses each available answer received.

The use of a quantitative software tool to support the data analysis of the novice questionnaire was considered, and several options were examined including SPSS (SPSS Inc. 2006). However the decision was taken to use a manual approach due to the relatively small volume of data collected, and the additional effort required to perform the data analysis using a specialised quantitative software tool.

3.4.3 Results

For the sake of consistency in the analysis and presentation, the subsections below each represent a section in the questionnaire (as shown in Table 3.4.1), and contain a summary of the questions asked followed by an analysis of the responses.

3.4.3.1 General Information

In Section 1 (General Information) of the questionnaire, respondents were asked about their experience with respect to requirements elicitation, including the number of years of practice they have had, and the number of projects they have been involved in. We also asked them about the typical duration of these projects, and their size in terms of the number of people involved. Respondents were then asked what their typical role was within these projects, and how they had learnt to elicit requirements for software systems. The specific questions and answers from this section of the questionnaire are shown below.

Question 1: How many years experience do you have in projects eliciting requirements for software systems?

Available Responses	*Number of Responses*	*Total Percentages*
1	5	21.7 %
2	6	26.1 %
3	3	13.0 %
4	1	04.4 %
Almost 5	8	34.8 %
Total:	23	100.0 %

Question 2: How many projects have you been involved in the eliciting of requirements for software systems?

Available Responses	*Number of Responses*	*Total Percentages*
1-3	9	39.1 %
4-6	4	17.4 %

7-9	4	17.4 %
10-12	0	00.0 %
More than 12	6	26.1 %
Total:	23	100.0 %

Question 3: What is the typical duration in months of these projects that you have been involved in?

Available Responses	*Number of Responses*	*Total Percentages*
Less than 6	14	60.9 %
6-12	7	30.4 %
13-24	2	08.7 %
25-36	0	00.0 %
More than 36	0	00.0 %
Total:	23	100.0 %

Question 4: What is the typical size of these projects in terms of the number of full time people involved in them?

Available Responses	*Number of Responses*	*Total Percentages*
Less than 5	6	26.0 %
5-10	15	65.2 %
11-25	1	04.4 %
26-50	0	00.0 %
More than 50	1	04.4 %
Total:	23	100.0 %

Question 5: Which title best describes your typical role in these projects?

Available Responses	*Number of Responses*	*Total Percentages*
Analyst	4	17.4 %
Consultant	2	08.7 %
Architect	0	00.0 %
Designer	1	04.4 %
Programmer	9	39.1 %

Developer	1	04.4 %
Software Engineer	0	00.0 %
Systems Administrator	0	00.0 %
Project Manager	6	26.0 %
Total:	23	100.0 %

Question 6: How did you primarily learn to elicit requirements for software systems?

Available Responses	***Number of Responses***	***Total Percentages***
Self taught (e.g. text books)	1	04.4 %
Through experience	5	21.7 %
University or TAFE course	12	52.1 %
Professional training course	1	04.4 %
On the job training	4	17.4 %
From a mentor	0	00.0 %
By observing others	0	00.0 %
Total:	23	100.0 %

From the responses to this section of the questionnaire, we can see that the average experience of the respondents is approximately 3 years, with almost half (48%) of them having had 2 years or less, and more than a third (35%) with almost 5 years experience. On average each respondent had worked on at least 6 projects, and more than a third (39%) have worked on 3 or less, with more than a quarter (26%) having worked on 12 or more projects. The average typical duration of a project was approximately 6 months, with more than 90% of them lasting less than 12 months, and 100% lasting less than 2 years in duration.

The typical size of the average project team was approximately 8 people, with over 90% of them having 10 members or less, and only two respondents typically working in project teams of more than 10 people. The most common designation for the respondents in these project teams was Programmer (39%), with Project Manager (26%) being the second most common. Only four of the respondents (17%) were actually assigned the role of Analyst when performing requirements elicitation for

software systems. More than 50% of the respondents learnt how to perform requirements elicitation at University or Tafe (Technical College), and of the remainder about half (22%) learnt through experience, and the other half (17%) via on the job training.

3.4.3.2 State of Practice

In Section 2 (State of Practice) of the questionnaire, respondents were asked about the number of projects they have been involved in which had problems related to poor requirements. They were also asked to rate their personal performance when eliciting requirements, and the overall level of requirements elicitation performance in industry today. We then asked them about the type of issues they have encountered, and what level of process guidance they had use for requirements elicitation in projects. The specific questions and answers from this section of the questionnaire are shown below.

Question 7: How many of the projects you have been involved in have had problems related to poor requirements before and/or after the delivery of the software?

Available Responses	*Number of Responses*	*Total Percentages*
None	0	00.0 %
Few	10	43.5 %
Some	10	43.5 %
Most	3	13.0 %
All	0	00.0 %
Total:	23	100.0 %

Question 8: How would you rate your personal performance in eliciting requirements for the projects you have been involved in?

Available Responses	*Number of Responses*	*Total Percentages*
Very poor	0	00.0 %
Poor	9	39.1 %

Fair	3	13.0 %
Good	10	43.5 %
Very good	1	04.4 %
Total:	23	100.0 %

Question 9: What do you think is the overall level of performance in requirements elicitation by industry today?

Available Responses	*Number of Responses*	*Total Percentages*
Very poor	0	00.0 %
Poor	18	78.2 %
Fair	4	17.4 %
Good	1	04.4 %
Very good	0	00.0 %
Total:	23	100.0 %

Question 10: How often have you encountered the following issues during requirements elicitation?

	Never (0)	Rarely (1)	Some (2)	Often (3)	Always (4)	***Score (/92)***
Not all of the relevant stakeholders have been identified	1 (0)	9 (9)	9 (18)	4 (12)	0 (0)	***39***
Not all of the relevant stakeholders have been involved	0 (0)	10 (10)	3 (6)	10 (30)	0 (0)	***46***
The stakeholders do not know exactly what they want/need	0 (0)	1 (1)	11 (22)	7 (21)	4 (16)	***60***
The stakeholders can not clearly describe what	0 (0)	1 (1)	8 (16)	12 (36)	2 (8)	***61***

they want/need						
The stakeholders are not committed or cooperative	0 (0)	12 (12)	4 (8)	6 (18)	1 (4)	***42***
The stakeholders ask for unrealistic or non-feasible requirements	1 (0)	1 (1)	13 (26)	7 (21)	1 (4)	***52***
The requirements are not well understood	0 (0)	6 (6)	6 (12)	10 (30)	1 (4)	***52***
The requirements are not correctly documented	0 (0)	2 (2)	13 (26)	5 (15)	3 (12)	***55***
Requirements are missing or incomplete	0 (0)	10 (10)	3 (6)	9 (27)	1 (4)	***47***
Requirements are incorrect	0 (0)	10 (10)	10 (20)	3 (9)	0 (0)	***39***
Some requirements conflict with other requirements	0 (0)	10 (10)	7 (14)	5 (15)	1 (4)	***43***

Question 11: What level of process guidance do you use for requirements elicitation in these projects?

Available Responses	***Number of Responses***	***Total Percentages***
None (i.e. ad-hoc or random)	6	26.1 %
Basic (i.e. high level overview)	14	60.9 %
Defined (i.e. detailed steps and tasks)	3	13.0 %
Total:	23	100.0 %

From the responses in this section of the questionnaire, we can see that approximately 56% of the respondents stated that some or most of the projects they have been involved in had problems as a result of poor requirements, with the remaining ten of the twenty-three respondents saying a few, and no respondents selecting none or all.

More than 50% of the respondents rated their performance when eliciting requirements as fair or poor, with only 44% of the total respondents rating theirs as good. Only one respondent considered their performance as very good, and no respondent believed their performance was very poor. Over 95% of the respondents stated that the overall level of performance in requirements elicitation by industry today was at best fair, with more than 80% of those saying it was poor. Only one respondent out of the twenty-three believed that the general state of performance in industry for requirements elicitation was good or better.

It is interesting to note that although the vast majority of the respondents said they only had problems with requirements on some or few projects, and almost half rated their performance in eliciting requirements as good or better, over three quarters of the respondents rated the overall level of performance of requirements elicitation in industry as poor. This would tend to imply that there is somewhat of a discrepancy between the personal experience of the respondents, and the general state of practice as perceived by them. This may be the result of either the respondents being overly generous with respect to rating their own individual performances, or the respondents being too critical in their perceptions of the industry performance in general.

The most common issue encountered by the respondents, from a list of common issues found in the literature, was stakeholders not being able to clearly describe what they want/need, closely followed by stakeholders not knowing exactly what they want/need. The respondents stated that the issues least often encountered were not identifying all the relevant stakeholders, and the requirements being incorrect. All but two of the eleven listed issues had been encounter by all of the respondents, with all but one happening at least some of the time to more than 50% of the respondents. More than 60% representing fourteen respondents used basic high-level process guidelines when performing requirements elicitation, and more than a quarter (26%) not using any at all. Only three of the twenty-three respondents used defined process guidance in projects when eliciting requirements for software systems.

3.4.3.3 Techniques and Tools

In Section 3 (Techniques and Tools) of the questionnaire, respondents were asked how often they used which techniques and tools to elicit requirements within project for the development of software systems. They were then asked what were the important factors taken into consideration when deciding which of these techniques and/or tools were to be used. The specific questions and answers from this section of the questionnaire are shown below.

Question 12: How often do you use the following techniques for requirements elicitation?						
	Never (0)	Rarely (1)	Some (2)	Often (3)	Always (4)	***Score (/92)***
Interviews	0 (0)	3 (3)	3 (6)	12 (36)	5 (20)	***65***
Questionnaires	3 (0)	3 (3)	14 (28)	3 (9)	0 (0)	***40***
Modeling (e.g. UML, DFDs, ERDs)	0 (0)	5 (5)	10 (20)	3 (9)	5 (20)	***54***
Scenarios (e.g. Use Cases, Storyboards)	0 (0)	4 (4)	8 (16)	8 (24)	3 (12)	***56***
Viewpoint-oriented	6 (0)	5 (5)	5 (10)	7 (21)	0 (0)	***36***
Goal-directed	1 (0)	10 (10)	7 (14)	4 (12)	1 (4)	***40***
Prototypes	0 (0)	2 (2)	4 (8)	13 (39)	4 (16)	***65***
Brainstorming	0 (0)	6 (6)	1 (2)	6 (18)	10 (40)	***66***
Group Workshops	0 (0)	0 (0)	4 (8)	4 (12)	15 (60)	***80***
Observation	1 (0)	9 (9)	7 (14)	3 (9)	3 (12)	***44***
Apprenticing	6 (0)	14 (14)	0 (0)	3 (9)	0 (0)	***23***

Question 13: How often do you use the following tools for the elicitation of requirements?						
	Never (0)	Rarely (1)	Some (2)	Often (3)	Always (4)	***Score (/92)***
Word processors (e.g. Word)	0 (0)	1 (1)	0 (0)	0 (0)	22 (88)	***89***
Spreadsheets (e.g. Excel)	0 (0)	0 (0)	7 (14)	0 (0)	16 (64)	***78***
Databases (e.g. Access)	2 (0)	3 (3)	6 (12)	3 (9)	9 (36)	***60***
Document templates (e.g. IEEE)	3 (0)	1 (1)	15 (30)	2 (6)	2 (8)	***45***
Requirements Management tools (e.g. RequisitePro)	8 (0)	5 (5)	10 (20)	0 (0)	0 (0)	***25***
Diagramming tools (e.g. Visio)	2 (0)	0 (0)	6 (12)	10 (30)	5 (20)	***62***
Modelling tools (e.g. UML Toolkit)	3 (0)	7 (7)	9 (18)	3 (9)	1 (4)	***38***
White boards and flip charts	0 (0)	0 (0)	3 (6)	14 (42)	6 (24)	***72***
Audio and video recorders	7 (0)	4 (4)	9 (18)	3 (9)	0 (0)	***31***

Question 14: How important are the following factors when you decide which requirements elicitation technique and/or tool to use?						
	Not important at all (0)	Not very important (1)	Somewhat important (2)	Important (3)	Very important (4)	***Score (/92)***
Familiarity	0 (0)	1 (1)	5 (10)	16 (48)	1 (4)	***63***
Past success	0 (0)	0 (0)	3 (6)	14 (42)	6 (24)	***72***
Speed	0 (0)	0 (0)	6 (12)	11 (33)	6 (24)	***69***
Effort	0 (0)	0 (0)	8 (16)	8 (24)	7 (28)	***68***
Cost	0 (0)	1 (1)	10 (20)	7 (21)	5 (20)	***62***
Flexibility	0 (0)	1 (1)	1 (2)	20 (60)	1 (4)	***67***
Usefulness	0 (0)	0 (0)	3 (6)	12 (36)	8 (24)	***66***
Ease of use	0 (0)	0 (0)	1 (2)	17 (51)	5 (20)	***73***
The customer's preferences	0 (0)	5 (5)	5 (10)	12 (36)	1 (4)	***55***
Your organisation's preferences	0 (0)	3 (3)	2 (4)	10 (30)	8 (24)	***61***

From the responses in this section of the questionnaire, we can see that Apprenticing and Viewpoint-oriented techniques were the least used by the respondents from the list, followed by Questionnaires, Goal-directed, and Observation. Group Workshops was by far the most widely used technique for requirements elicitation, ahead of Interviews, Brainstorming, and Prototypes. In fact fifteen of the twenty-three respondents (65%) always used Group Workshops, with four using them often, and the remaining four using them sometimes.

Twenty-two respondents out of the twenty-three always used Word Processors during requirements elicitation, with Spreadsheets, and White boards and flip charts also being widely used. Of all the requirements elicitation tools listed, Requirements Management applications and Audio and video recorders were by far the least used.

Databases were more widely used by the respondents than Document templates, and Diagramming tools were significantly more often used than those for Modelling.

Ease of use was deemed the most important factor when deciding which requirements elicitation techniques an/or tool to use, followed by a group of factors with similar levels of importance including Past success, Speed, Effort, Flexibility, and Usefulness. The least most important factors considered by the respondents were the customer's preferences and the respondents' organisation preferences. The factor thought important in deciding which technique and/or tool to use by more respondents than any other was Flexibility.

3.4.3.4 Assistance and Support

In Section 4 (Assistance and Support) of the questionnaire, respondents were asked about the types of assistance they believed would help improve their performance during requirements elicitation, and how useful they believed tool support would be for various requirements elicitation activities. The specific questions and answers from this section of the questionnaire are shown below.

Question 15: How often do you think the following types of assistance would help you improve your performance during requirements elicitation?						
	Never (0)	Rarely (1)	Some (2)	Often (3)	Always (4)	***Score (/92)***
Process guidelines	0 (0)	0 (0)	5 (10)	13 (36)	5 (20)	***66***
Tool support	0 (0)	1 (1)	5 (10)	16 (48)	1 (4)	***63***
Increased education and training	0 (0)	0 (0)	1 (2)	14 (42)	8 (32)	***76***
Better elicitation techniques	0 (0)	0 (0)	12 (24)	6 (18)	5 (20)	***62***
More experience	0 (0)	6 (6)	0 (0)	6 (18)	11 (44)	***68***
Additional time	0 (0)	0 (0)	5 (10)	12 (36)	6 (24)	***70***
Bigger budget	0 (0)	9 (9)	4 (8)	9 (27)	1 (4)	***48***

Extra resources	0 (0)	1 (1)	3 (6)	14 (42)	5 (20)	***69***

Question 16: How useful do you think tool support would be for the following requirements elicitation activities?						
	Not useful at all (0)	Not very useful (1)	Some what useful (2)	Useful (3)	Very useful (4)	***Score (/92)***
Guidance through the process	0 (0)	0 (0)	7 (14)	13 (36)	3 (12)	***62***
Understanding the operating environment	0 (0)	0 (0)	6 (12)	14 (42)	3 (12)	***66***
Identifying stakeholders	0 (0)	1 (1)	7 (14)	14 (42)	1 (4)	***61***
Managing the communication with stakeholders	0 (0)	1 (1)	10 (20)	10 (30)	2 (8)	***59***
Recording and storing requirements	0 (0)	0 (0)	10 (20)	10 (30)	3 (12)	***62***
Managing and changing requirements	0 (0)	0 (0)	13 (26)	5 (15)	5 (20)	***61***
Elicitation techniques selection	0 (0)	1 (1)	16 (32)	3 (9)	3 (12)	***54***
Using different elicitation techniques	0 (0)	1 (1)	10 (20)	7 (21)	5 (20)	***62***
Managing and running interviews	0 (0)	11 (11)	5 (10)	2 (6)	5 (20)	***47***
Managing and running workshops	0 (0)	5 (5)	10 (20)	3 (9)	5 (20)	***54***

From the responses in this section of the questionnaire, we can see that increased education and training was believed to be the type of assistance that would most improve requirements elicitation performance, ahead of additional time, extra resources, and more experience. These were followed by process guidelines, tool

support, and better elicitation techniques. A bigger budget was considered relatively unlikely to improve performance, but none of the types of assistance listed was considered never to be likely to improve performance by any of the respondents.

Despite tool support receiving the most responses for being able to often improve performance during requirements elicitation, it achieved only the sixth highest score in the total rankings for the eight types of assistance listed. This under-perceived need for tool support during requirements elicitation from the respondents may be the direct result of several factors. As we saw from the answers to Question 13, most of the respondents mostly used very simple tools during requirements elicitation, such as Word processors and Spreadsheets. It is therefore possible that the respondents could not see how other tools along the same lines would be able to improve their performance during requirements elicitation significantly. It is also possible that the respondents are either unaware or unfamiliar with most of the available requirements elicitation tools, and were therefore unable to conceptualise and visualize how a tool specifically designed and developed to support requirements elicitation could be useful and provide a major benefit to them.

Understanding the operating environment was thought to be most useful activity for tool support, followed by guidance through the process, recording and storing requirements, and using different elicitation techniques. Managing and running interviews was believed to be the activity where tool support would be the least useful with eleven of the twenty-three respondents stating it would be not very useful. At least one respondent was of the opinion that tool support would be very useful for each of the activities listed except identifying stakeholders. Furthermore, tool support was not considered to be not useful at all by any of the respondents for any of the listed activities.

3.4.4 Discussion

The results from the questionnaire show that novices more often work in smaller teams on projects of relatively short durations. It is interesting that only a small number of the novices were actually recognised as being analysts, with requirements

elicitation being just one of the many tasks assigned to them as either a Programmer or Project Manager in most cases. This shows that requirements elicitation and more generally RE is rarely conducted by a dedicated resource, with only that activity as their sole and separate responsibility. It is important to note that almost half of the novices have not had any formal training in requirements elicitation despite it being part of their job, supporting the perceived lack of expertise in current practice.

Most of the novices regularly encountered problems in projects because of poor requirements. This is not surprising given that more than half rated their own performance less than Good, and the majority rating the overall industry performance as poor. This highlights not only the troubling current state of affairs in requirements elicitation practice, but also the real and existing need for significant improvement. Stakeholder related issues seemed to be those occurring most often, which identifies the human-centric nature of eliciting requirements for software systems. Furthermore the regularity and range of issues also shows how problematic the activity as a whole really is. Just as concerning is the fact that more than a quarter of the novices do not use any process guidance at all when performing requirements elicitation.

Among novices there still appears to be a tendency to mainly use the more traditional requirements elicitation techniques, and in particular group workshops and interviews. This shows that in general novices are not using many of the requirements elicitation specific techniques, and especially those developed more recently. Likewise, the tools mostly used by the novices tended to be those that are generic and basic in nature, such as word processes and spreadsheets, and once again those specific to requirements elicitation are not widely used at all. The fact that novices considered ease of use and flexibility as the most important factors when deciding which techniques and/or tools to use is not surprising given their limited experience, expertise, and education. The little importance shown to their organisations' and customers' preferences may represent a willingness by novices to try new and different techniques if attractive, effective, and efficient.

It is little wonder given the fact that the respondents where by definition novices, and almost half has not formally learnt requirements elicitation, that education and training was deemed the most important aspect for improving requirements elicitation

performance. It was also seen that novices seek improvement through better implementations of skills and methods rather than via bigger budgets. The fact that the types of assistance listed were all believed to be of some benefit for improving requirements elicitation practice once again highlights the need for continuing and additional work in this area. Domain understanding was suggested as the activity that would most benefit from tool support, which is not unexpected given that the context in which requirements elicitation takes place is never exactly the same twice. However, more important was that in general the novices felt tool support could be of some advantage for all of the different activities listed that makes up the entire process of requirements elicitation.

Limitations

Many of the restrictions mentioned with respect to the expert interviews, also apply to our use of an online questionnaire aimed at novices. Once again the opportunity for follow-up questions, response clarification, and potentially useful further in-depth discussion do not exist when using an anonymous web-based questionnaire. However the simplicity of the questions and the ease with which respondents were able to answer them somewhat reduces this disadvantage as a trade-off. Likewise the number of respondents and the detail of the questions could also be considered as restrictions on the research, but conducting a strict pilot study has also reduced this. Once more the memory and opinions of the respondents may be overly affected by their more recent experiences, or by their most significant successes and failures. In order to minimize these types of biases, invitations to participate in the questionnaire were not sent to two people that work together, or that had worked for the same organisation at any time in the past.

3.4.5 Section Summary

In summary, the key points identified in this section are:

1. It was often necessary for the novices to elicit requirements whilst performing other roles in the project including that of programmer and project manager

2. Most of the analysts had regularly worked on projects with problems caused by poor requirements

3. The vast majority of the novices rated both their own performance, and that of industry overall, in requirements elicitation as generally not good

4. A range of significant issues was commonly encountered by most of the novices, and most often relating to the stakeholders

5. For the most part the novices only used some of the more traditional techniques for elicitation, and in particular group workshops and interviews

6. Only generic tools were used by the novices during requirements elicitation, and their decision was based primarily on ease of use and flexibility

7. The novices were generally of the opinion that increased education and tool support would lead to improvements in practice for most elicitation tasks

3.5 Chapter Summary

The research presented in this chapter was focused on reviewing the available literature on the current state of practice in requirements elicitation, and through experts interviews and a novice questionnaire, either confirming or refuting some of the more commonly held perceptions. The interviews and questionnaire also represented requirements elicitation sessions, and together with the results from our review of requirements elicitation theory, form the basis of a new and improved approach and tool described in the following two chapters respectively. In addition, the results from this and the previous chapter also support the core list of approach and tool requirements detailed in Appendix E. It is important to remember that yet again we have focused specifically on requirements elicitation and not the entire RE process or software development life cycle. Nor have we investigated some of the more finer grained activities often associated with requirements elicitation such as negotiation and prioritisation.

If we look at the literature on requirements elicitation practice, and the results from the expert interviews, we can see that both agree on the current poor state of requirements elicitation practice, and the need for greater attention despite recent progress. The findings from the novice questionnaire also sided with the literature that the performance of requirements elicitation by industry today was in need of significant improvement. The apparent gaps between theory and practice, and expert and novice analysts presented in the literature are supported by the calls from the experts for more training of practicing analysts, and an increase in the development and usage of tool support. These positions were further confirmed by the heavy dependency of the questioned novices on primarily traditional techniques and generic tools, and by the clearly demonstrated belief that additional and better training and tools would improve their performance. Of the substantial range of issues that are often mentioned in the literature as frequently occurring during requirements elicitation in practice, both the novices and experts agreed that those relating to problems with the stakeholders were by far the most common.

We can therefore see that for the most part the overall results from both the expert interviews and novice questionnaire are as expected, and support the trends identified in the literature, consistent with commonly held perceptions about requirements elicitation practice. However it appears that novices do not place the same level of importance on having a clearly defined process when performing requirements elicitation as does the literature and the experts in the field we interviewed. Although the novices and experts agreed that tools are potentially useful in improving requirements elicitation practice, novices in general saw a greater scope for their implementation and usage in the process than did the experts.

We have seen in this chapter that in practice the activity of eliciting requirements for software systems is an important and difficult multifaceted process (Chatzoglou 1997), which is particularly problematic due to its criticality, complexity, the number of roles the analyst must play, and the range of issues commonly encountered. By nature requirements elicitation is an imperfect and imprecise activity, however as we have also seen, the production of high quality requirements through effective elicitation is essential for the engineering of successful software products and projects.

CHAPTER 4: The OUTSET Approach

4.1 Chapter Overview

In the two previous chapters we performed a review of the theory, and a survey of practice, on and around the subject of requirements elicitation for software systems development. As a result, we were able to identify that there is currently a real and perceived need for the development and evaluation of new and improved approaches for requirements elicitation, which reduce the complexities whilst directly addressing some of the commonly occurring issues and challenges often encountered in practice.

In this chapter we will detail the OUTSET approach to support the early stages of requirements elicitation in software development, based on guided collaborative workshops and situational method engineering (Section 4.2). We will describe the construction of a method using the meta-model (Section 4.3) and process framework (Section 4.4) of the approach. We will then present an example implementation of the OUTSET approach for a real world project (Section 4.5). This will be followed by a discussion (Section 4.6) and summary (Section 4.7) of the entire chapter.

The purpose of this chapter is therefore to provide an approach for performing the process of requirements elicitation, which can be readily applied to real-world projects by practitioners (**Research Goal 3**). We also aim to offer researchers an example of how situational method engineering can be applied to very practical activities and situations in the software development process in order to improve both the process and the results (*Research Question 8*). The overriding intention of the approach, however, is to support novice analysts structure requirements elicitation workshops based on project specific characteristics.

4.2 Background

Method Engineering (ME) represents a structured way in which methods for software development activities, such as requirements elicitation, can be designed, constructed, and adapted. In this way methods are assembled from multiple individually identifiable parts, often referred to as ‘method fragments’ or ‘method chunks’. Situational Method Engineering (SME) is therefore the configuration of these resultant methods, specifically for individual projects (Brinkkemper 1996). Naturally this is an important topic as no two software development projects are exactly the same, and all projects cannot be adequately supported by a single static method. This is especially the case with requirements elicitation, where the heavy dependence on human involvement adds a significant number of social and communication variables.

The idea of creating a situational method for requirements elicitation is not new. As early as 1982, Davis identified the need and importance of developing situational requirements elicitation methods (Davis, G. B. 1982), although he referred to them as “requirements determination strategies”. The basic premise, however, of characterizing a specific project based on some criteria, and as a result, selecting from a set of methods those that are most appropriate, remains the same. It is important to note that Davis also suggests and stresses that the development of such a method should be based on the types of information to be elicited.

The broader scope of ME research for software development in recent times has included models for situational method engineering (Ralyté, J., Deneckère & Rolland 2003), assembly techniques and rules for method construction (Brinkkemper, Saeki & Harmsen 1999; Ralyté, J. & Rolland 2001), and more generic process modelling and engineering approaches (Henderson-Sellers 2002). The combined result of this work over the past ten years in particular, in conjunction with the concepts recommended by Davis described above, provides a suitable foundation for the development of a new activity specific situation method as presented in this chapter for requirements elicitation. Of specific interest and relevance to the objectives of our research is the work by Henderson-Sellers and Firesmith relating to the OPEN Process Framework (Firesmith & Henderson-Sellers 2001), which has been applied to the entire software

development lifecycle, as opposed to just the elicitation of requirements as is our case. Although much broader in scope, the OPF has similar goals to our approach, and is likewise process oriented.

It can be seen that requirements elicitation alone represents an excellent candidate for the implementation of a situation specific method. As it is often the first phase of a development project, it does not have to conform to the assumptions or constraints of other methods employed in previous phases, or rely on the outputs from other activities. Furthermore, the only way to address situation specific problems and issues during requirements elicitation, of which there are many, is to have a tailorable method and flexible process for its execution. On the other hand, at the requirements elicitation stage, very little may actually be known about the project and the target system, making it difficult to determine which situation characteristics exist and should be addressed.

The name ‘OUTSET’ was chosen for our approach because with it we aim to support requirements elicitation, being at the beginning of software development, and novice analysts, being in the early years of their careers. The approach is intended to be used to structure and conduct collaborative requirements elicitation workshops, within projects for the development of software systems, using a combination of techniques, in association and cooperation with other requirements elicitation approaches and techniques. Furthermore, the approach primarily supports the early stages of requirements elicitation, i.e. information gathering, when the requirements are least understood, and the process is typically at its most undefined. The starting point of the approach, and the assumed motivation for the project, is to find a solution to a problem or to achieve a goal, and that the project inception has already taken place, leading to the next step being a project proposal, feasibility study, or requirements investigation.

4.3 Meta-levels of the Approach

The OUTSET approach can be explained using the three-tiered structure as seen in Figure 4.3.1 (adapted from (Henderson-Sellers 2002)), which differentiates between a method meta-model, a specific instantiation of that method meta-model, and an individual project specific instance of a specific instantiation. This can be described in more basic and practical terms by the following example. L2 represents the components and rules for constructing a requirements elicitation process for an organisation that produces different types of software. L1 then represents one possible example of a process, constructed using the components and rules set out in L2, for a specific type of project (e.g. the development of customized business information systems). L0 would then represent a project specific enactment of that process detailing the requirements, goals, and constraints of the business information system for that particular customer.

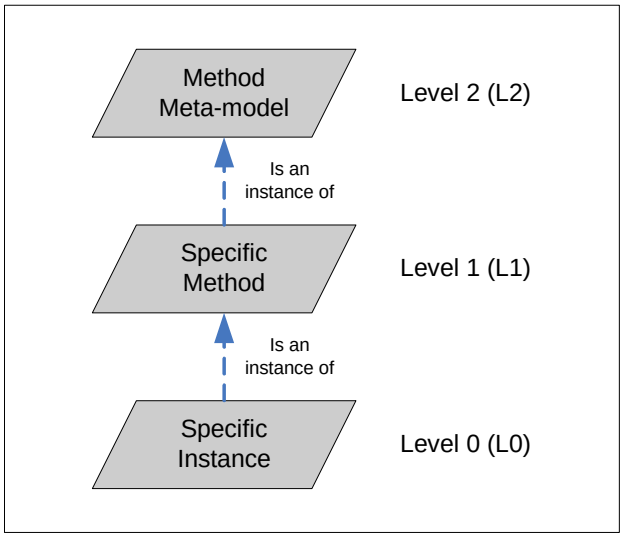

Figure 4.3.1: Approach meta-levels

The top level of this structure (L2), from which processes are constructed, consists of four main ‘meta-types’ or ‘classes’ as shown in Figure 4.3.2, and described in Table 4.3.1 below. In fact, instances of these meta-types (Info Types, Tasks, Sources, and Techniques), which can be considered as ‘method fragments’ or ‘method chunks’ in terms of the existing situational method engineering literature, are just the building

blocks used to construct what we call ‘method components’, and it is actually a set of these method components that are selected and sequenced to form a specific method (L1) which can then be enacted (L0).

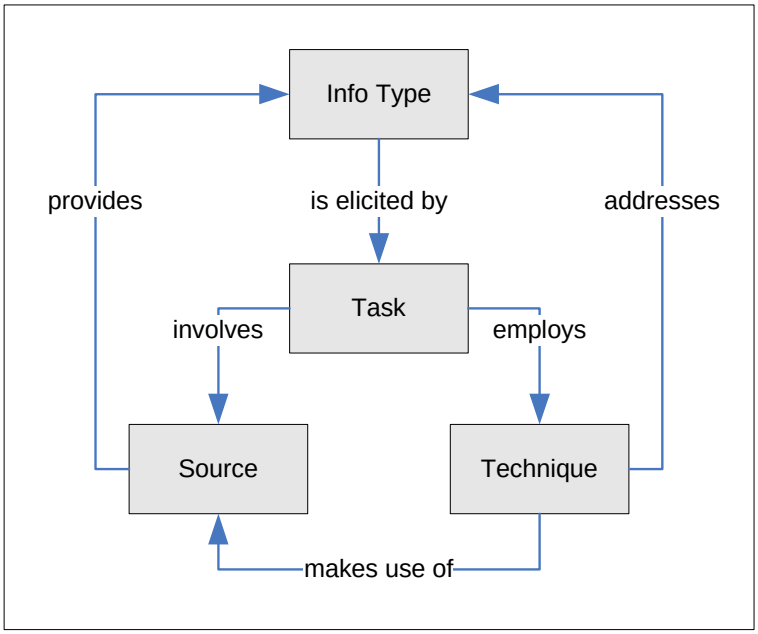

Figure 4.3.2: Method component meta-model

We hereby introduce and use the term ‘method component’ to describe a single instance of the entire method meta-model based on an individual task. Therefore, a method component is a method fragment of the Task class, with one or more related method fragments of each of the Info Type, Source, and Technique classes. In summary, a method component is a task that elicits one or more info types from one or more sources using one or more techniques. These method components are used as the base unit of work to organize and execute requirements elicitation within a workshop environment, by placing them into a structured sequence according to the process framework described in the next section.

Table 4.3.1: Method meta-model classes

Name	***Description***
Info Type	A specific kind of required data or knowledge, such as ‘project assumptions’ and ‘design constraints’.
Task	A specific individual unit of work. Examples include ‘identify

	key constraints' and 'define work processes'.
Source	A specific place or object that provides information. Sources can be individuals, groups, artefacts, and systems.
Technique	A specific way of performing a requirements elicitation task, such as by questionnaire or brainstorming.

4.4 Process Model for the Approach

The OUTSET approach presented in this chapter represents both a process for engineering a situational method, and a process for performing requirements elicitation. The process model for the entire approach as shown in Figure 4.4.1 (adapted from (Brinkkemper 1996)), provides an overview and illustrates that the engineering of the method and the process of elicitation itself begin with characterization of the project at hand, leading to the construction of the method for that project, and then execution of that method within the project at hand. These individual 'steps' of the OUTSET approach are subsequently detailed in the following subsections.

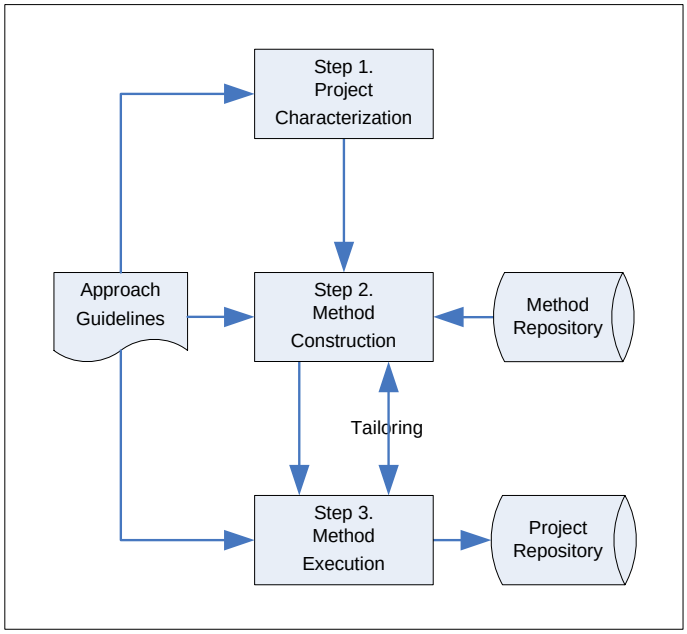

Figure 4.4.1: OUTSET approach process model

Step 1: Project Characterization

The first step of the OUTSET approach is Project Characterization. It is at this point that the situational characteristics of the project at hand are identified in order to direct the construction of the method to be used. There are a number of different ways to characterize a software development project based on the specific situation. One suggestion has been to use a taxonomy of specific problem, solution, and other project

situational factors (Hickey & Davis 2003c). Another has been to base the characterization on the goals, risks, opportunities, and challenges of the project (Kurtz 2001). Although these are useful ways of characterizing projects, this information is rarely available before the requirements elicitation phase, and in fact it is the actual purpose of requirements elicitation to discover and develop this information. Therefore, we use the following three basic attributes to categorize the specific requirements elicitation project:

1. **Definition** – The definition of the type of elicitation project being conducted. Examples of project definitions include Custom Development (i.e. Bespoke Software), COTS Selection, and Feasibility Study.

2. **Domain** – The general application domain of the envisaged system. Examples of application domains include Business Information, Group Support, and Embedded Control.

3. **Deliverable** – The required system related output from the elicitation project. Examples of deliverables include Requirements Specification, Concept of Operations, and Vision & Scope documents.

Although admittedly somewhat basic, characterization of the project by using only these three variables (hereafter referred to as the '3Ds') does however take into account three of the most important and influential situational elements that are usually known or can be assumed at this very early stage of the software development lifecycle, even by novice analysts. The combination of values for these 3Ds for a given project is used to guide the construction of the method as described next.

Step 2: Method Construction

The second step of the OUTSET approach is Method Construction. It is at this point that the method fragments are selected from the Method Repository and assembled into method components. These project components are then structured and sequenced according to the process framework. These operations are referred to collectively as the method construction (Brinkkemper, Lyytinen & Welke 1996), the result of which is an executable method than can then be enacted for the project at

hand. Significant support has been included for this step within the approach, as the development of methods from predefined components can be difficult for novices and even experts.

The Method Repository mentioned above can be represented as a series of lookup tables, which detail instances of the different meta-types (called method fragments) and their relationships to other method fragments. There are ten tables in total, being one flat list for each of the meta-model classes (see Table 4.3.1), and one for each of the different possible inter-class relationships. In the Info Type and Task flat lists, flags are used to identify which ones are recommended for each of the possible values for the 3Ds identified in Step 1 of the approach. An analyst can therefore ensure that all the appropriate Info Types are addressed, and all the prudent Tasks are included in the requirements elicitation method, by using these tables to assemble a set of method components.

Once the method fragments have been selected and assembled to form a set of method components, this set of method components must then be structured and sequenced to complete the construction of the requirements elicitation method. In order to support this operation, the approach guidelines include a process framework that can be used by the analyst as a template for arranging and ordering the method components. This process framework, as can be seen in Figure 4.4.2 below, divides the complete set of method components into three key workshop types or phases of ‘Scoping’, ‘High-level’ and ‘Detailed’. These phases are then divided into three stages being ‘Preparation’, ‘Performance’, and ‘Presentation’. Method components of the Performance stage of each phase are subjected to a third division into five different areas of interest. This hierarchical division of phases and stages has been based on a combination of sources from the literature including (Gottesdiener 2002) and (Robertson, S. & Robertson 1999).

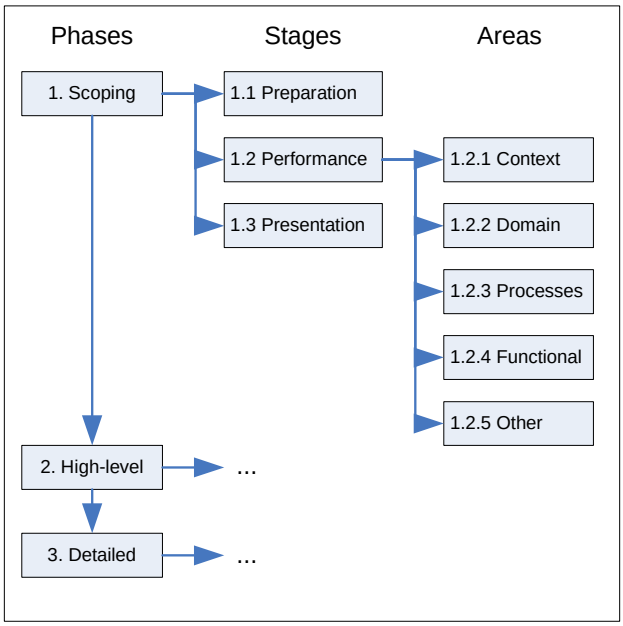

Figure 4.4.2: OUTSET approach process framework

It is anticipated that more experienced analysts would be able to select instances of each class from the Method Repository, assemble method components, and construct the resultant method with little or no reference to the approach guidelines and process framework. Furthermore, expert analysts often know familiar and proven ways of doing things during requirements elicitation, and therefore would much prefer the freedom to choose how the method fragments and method components are to be used. Novice analysts, on the other hand, would undoubtedly prefer a more prescribed approach to method construction, given that they by definition lack the knowledge of experience and range of expertise required to do this independently.

In direct response to this, and in order to satisfy the key goal of this research to support novice analysts and projects without a defined requirements elicitation process, we have developed and incorporated into the approach a concept that we have called ‘ready-made’ requirements elicitation methods. These represent pre-selected, pre-assembled, and pre-constructed methods for typical, common, and often occurring combinations of the 3Ds in practice. That is to say that the Method Repository can be pre-populated, and method components assembled, from either references found in the available literature, or from real world projects and supporting documentation. A basic example of one such possible ‘ready-made’ method is presented in Section 4.5 of this chapter.

Step 3: Method Execution

The third and final step of the OUTSET approach is Method Execution. Once the method has been constructed in accordance with the process framework using method components made up of method fragments from the Method Repository, the requirements elicitation part of the project can then be executed, with the results of the method components being stored in the Project Repository. The Project Repository can be represented as a set of detailed templates for the resultant different types of information elicited during the process.

During execution, each method component Task of the method is addressed utilizing the associated Techniques to elicit the required Info Types from the available Sources in support of, or within, a workshop environment. Each of the Performance stages prescribed by the process framework are facilitated by the analyst, and may be completed over a number of workshop sessions depending on the complexity of the project, and the accessibility of relevant stakeholders. It is probable that the same Info Type may be addressed by more than one Task at different stages of the method. In these cases, the level of detail investigated and the attributes elicited for the Info Type are usually different. Normally, each Task has at least one available Technique, and likewise each Info Type has at least one possible Source.

All method components can be repeated, removed, or reconfigured dynamically during the project by the analyst based on preferences or changing situations. This is referred to as ‘tailoring’ the method. Tailoring can take place either before the method is enacted, as in the case of adding or removing method components after construction, or during the execution of the process itself, such as selecting which of the requirements elicitation techniques to use for a given task. Like those constructed from scratch, ready-made methods can be tailored by modifying the Info Types, Tasks, Techniques, and Sources in the Method Repository before or after construction. This is important as it enables organisations and individuals to develop their own ‘template’ situational methods for different sets of project characteristics. It also creates a feedback and validation mechanism into the approach, adding a further level of flexibility and customization. An example of dynamic tailoring would be if the analyst believes insufficient information has been elicited for a particular Info

Type during a session, a new Task or Technique for that Info Type can be selected from the repository, and added to the existing method, or utilized there and then.

4.5 OUTSET Approach in Action

The following section provides a basic example of how an existing 'ready-made' method could be implemented and enacted in a real-world project as a basis for a novice analyst, within the context of the general requirements elicitation approach presented in this chapter. This instructional generic example aims only to illustrate how the basic principles of the approach can be applied to the fundamental areas of a requirements elicitation project, and was based on the results from our Literature Review (Chapter 2) and Survey of Practice (Chapter 3), in addition to various industry standards and a selection of project artefacts.

Step 1: Project Characterisation

The example ready-made method has the following situation project characteristics, i.e. values for the 3Ds, as defined in the project initiation phase (preliminary planning), and identified by the participating analyst:

Definition Type

The Definition Type for this example ready-made method, i.e. the definition of the type of elicitation project being conducted, is **Custom Software Development**. This represents a project where a software system is built to order according to the specifications of a particular customer or client.

Domain Type

The Domain Type for this example ready-made method, i.e. the general application domain of the envisaged system, is **Business Information System**. This means an information system that supports the business processes and operations of a specific organisation.

Deliverable Type

The Deliverable Type for this example ready-made method, i.e. the required system related output from the elicitation project, is **Software Requirements Specification**. This corresponds to a detailed description of the desired behaviour, which the target system should exhibit.

Step 2: Method Construction

The following subsections detail the Info Types, Tasks, Sources and Techniques recommended for the 3Ds combination mentioned above in the Project Characterisation step by this example ready-made method presented. All of these info types, tasks, sources, and techniques have been identified from the available literature and project documentation, as being both relevant and appropriate for the elicitation of requirements, for the above mentioned combination of the 3Ds.

Info Types

Normally at this point in the approach, the analyst decides what information is required to be elicited based on the Definition, Domain and Deliverable types. However, ready-made methods, such as the example presented here, provide the analyst with an initial set of necessary Info Types preconfigured to the results from the 3Ds in the Project Characterization step, as can be seen in the table below. In this case, the analyst is still free to add, change, and delete any of the suggested Info Types in accordance with the specific characteristics of the particular project at hand.

Table 4.5.1: Recommended Info Types

Name	*Description*
Project Information	Problem, mission, vision, and scope of the project
Deliverable Information	Desired result of the process, its audience, objectives, and overview
System Information	Background, perspective, context, and scope of the system
Goals	Objectives of the business with respect to the project and the system
Assumptions	Underlying assumptions upon which the project and system are based

Constraints	Overriding constraints that must be applied to the project and system
Environmental Details	Social and physical environmental details of the project and the system
Opportunities	Possible opportunities for improvements to be delivered by the system during the project
Challenges	Possible challenges which may be encountered during the project related to the system
Risks	Potential risks to the success of both the project and the system
Stakeholders	Stakeholders in the project, and sources of information related to the system
Work Processes	Detailed work process which the system should support
Functional Aspects	Features and functional requirements which should be provided by the system
Non-functional Aspects	Non-functional conditions and requirements which should be provided by the system
Implementation Details	Details relating to the implementation of target system including preliminary design solutions

Tasks

Next the analyst is required to choose which Tasks should be performed in order to elicit the required Info Types. The Tasks to be performed within a workshop environment suggested by this example ready-made method, in order to elicit the necessary Info Types previously identified, can be seen in the table below. Once again the analyst is allowed to add, change, and delete any of the recommended tasks in accordance with the specific characteristics of the particular project at hand.

Table 4.5.2: Tasks by Info Type

Info Type	***Tasks***
Project Information	Define the Problem, Need, and/or Idea behind the project and system Describe the Mission of the project related to the system Describe the Vision of the project related to the system Define what is considered in and out of scope for the project
Deliverable Information	Define the intended audience for the output of the elicitation project Define the objectives of the deliverable as they relate to the audience Describe an overview of the deliverable in terms of main sections
System Information	Describe the background of the system in terms of the original concepts Describe the perspective of the system with reference to other systems

	Define what is considered in and out of scope for the system
Goals	Define the main system goals and project drivers
Assumptions	Identify any and all assumptions that may affect the project Identify any and all system related assumptions within the project Describe in detail the system assumptions previously elicited
Constraints	Identify any and all constraints that may affect the project Identify any and all system related constraints within the project Describe in detail the system constraints previously elicited
Environmental Details	Describe the business environment of the project and system Identify the details of the environment in which the system will reside Describe the details of the system environment previously elicited
Opportunities	Identify any opportunities for improvement from the system Describe in detail the system opportunities previously elicited
Challenges	Identify any possible challenges for the system Describe in detail the system challenges previously elicited
Risks	Identify key risks that may affect the project and/or system Describe in detail the risk previously elicited and any others
Stakeholders	Identify key stakeholders that may be effected by the project and/or system Describe in detail the stakeholders previously elicited and others
Work Processes	Identify key work processes to be supported by the system Describe in detail the work processes previously elicited Describe the specific steps for the work process previously elicited
Functional Aspects	Identify the key functional aspects of the system Describe the key features and capabilities of the system Describe the specific functional system requirements
Non-functional Aspects	Identify the key non-functional aspects of the system Describe the key non-functional aspects of the system Describe the specific non-functional system requirements
Implementation Details	Identify the important details related to the implementation of the system Describe the important details related to the implementation of the system

Sources

The analyst is then typically required to identify the available sources from which to elicit the selected Info Types, using the chosen tasks, within a workshop environment. The Sources prescribed by the example ready-made method can once again be seen in the table below. It is important to note that not all sources, and particularity stakeholders, will always be available to participate in requirements elicitation

workshops, e.g. Competitors. However, it is assumed that the analyst will always be present for all information gathering events, even if only as a facilitator.

Table 4.5.3: Sources by Info Types

Info Type	*Sources*
Project Information	Customers, Sponsors, Project Managers
Deliverable Information	Customers, Sponsors, Project Managers
System Information	Customers, Sponsors, Project Managers
Goals	Customers, Sponsors, Project Managers
Assumptions	Project Managers, Business Analysts
Constraints	Project Managers, Business Analysts
Environmental Details	Existing Process, System Manuals, Company Literature Project Managers, Business Analysts, Managers, Supervisors, Domain Experts, System Administrators
Opportunities	Project Managers, Business Analysts, Supervisors
Challenges	Project Managers, Business Analysts, Supervisors
Risks	Project Managers, Business Analysts, Supervisors
Stakeholders	Project Managers, Business Analysts, Supervisors
Work Processes	Exiting Processes, System Manuals Project Managers, Business Analysts, Domain Experts, Supervisors, Key Users
Functional Aspects	Existing Process, System Manuals, Forms and Reports Project Managers, Business Analysts, Domain Experts, Supervisors, Key Users, Developers
Non-functional Aspects	Existing Process, System Manuals Project Managers, Business Analysts, Domain Experts, Supervisors, Key Users, Developers
Implementation Details	Project Managers, Business Analysts, Domain Experts, Supervisors, System Administrators, Developers

Techniques

After the Sources for the elicitation of the Info Types have been identified, and assigned to the corresponding selected Tasks, the analyst is now required to choose which techniques are to be used within the workshop environment. As we have seen previously in Chapters 2 and 3, the choice of elicitation technique can depend on a large number of factors, and not only the specific type of information that needs to be

elicited. For the example ready-made method presented here however, the proposed Techniques can be seen in the table below.

Table 4.5.4: Techniques by Info Types

Info Types	*Techniques*
Project Information	Interviews, Brainstorming, Document Analysis
Deliverable Information	Interviews, Brainstorming
System Information	Interviews, Brainstorming, Application Analysis
Goals	Interviews, Brainstorming, Goals
Assumptions	Interviews
Constraints	Interviews
Environmental Details	Interviews, Questionnaires, Document Analysis, Application Analysis, Domain Analysis
Opportunities	Interviews, Brainstorming
Challenges	Interviews, Brainstorming
Risks	Interviews, Brainstorming
Stakeholders	Interviews, Document Analysis, Application Analysis, Domain Analysis, Viewpoints
Work Processes	Interviews, Document Analysis, Application Analysis, Domain Analysis, Task Analysis, Use Cases, Scenarios
Functional Aspects	Interviews, Document Analysis, Application Analysis, Goals, Scenarios, Viewpoints, Questionnaires
Non-functional Aspects	Interviews, Document Analysis, Application Analysis, Goals, Questionnaires
Implementation Details	Interviews, Brainstorming

Step 3: Method Execution

In the Method Construction step described above, 39 tasks were suggested by the example ready-made method as necessary to elicit the required information types. Since each method component is based on an individual task, this means that there are a minimum of 39 core method components in the pre-selected, pre-assembled, and pre-constructed method for the identified 3Ds combination (i.e. the production of a Software Requirements Specification for the Custom Software Development of a Business Information System). These method components, within the context of the process framework, provide the basis for the Method Execution step of the approach,

and the performance of the collaborative and combinational requirements elicitation workshops as described by phase and stage in the subsections below.

The Scoping Phase

The most important part of the Preparation stage of the Scoping phase involves the task of identifying the relevant stakeholder sources for participation in the Scoping workshops. Typically these sources would include the Project Sponsors, i.e. the people supporting the project, Upper Management of the same organisation, and key members of the project team such as the Project Manager. Furthermore, any and all available external documentation sources relevant to the project should be studied, such as the marketing material and website of the target organisation. In addition, the participating analyst should carefully review and be familiar with the method components and sequence for the Scoping Performance stage, as well as gathering as much preliminary but relevant information as possible through informal discussions and observations.

During the Performance stage (25 method components as can be seen in Table 4.5.5 below), the participating analyst and stakeholders perform tasks to describe the problem, the mission, and the vision for the project, in addition to defining the boundaries for both the project and the system, i.e. what is in and out of scope. Furthermore, high-level goals for both the project and the system are established, and key assumptions and constraints are identified. It is also at this point that the major risks, opportunities, and challenges are identified, and stakeholders for the High-level and Detailed workshops are determined. A number of techniques are prescribed for these tasks including Brainstorming and Questionnaires.

Table 4.5.5: Scoping Performance Stage Method Components

Task	*Info Type*	*Source*	*Technique*
Business Context			
1. Define problem/need/idea 2. Describe project mission 3. Describe project vision 4. Define project scope	Project Information	Company Literature Customers Sponsors Project Managers	Interviews Brainstorming Document Analysis

5. Define deliverable audience 6. Define deliverable objectives 7. Describe deliverable overview	Document Information	Business Analysts	
8. Identify project assumptions	Assumptions		
9. Identify project constraints	Constraints		
10. Describe business environment	Environmental Details		
11. Identify system opportunities	Opportunities		
12. Identify system challenges	Challenges		
Application Domain			
13. Describe system background 14. Describe system perspective 15. Define system scope	System Information	System Manuals Project Managers Business Analysts Domain Experts	Interviews Questionnaires Document Analysis Application Analysis Domain Analysis
16. Define system goals	Goals		
17. Identify system assumptions	Assumptions		
18. Identify system constraints	Constraints		
19. Identify system environment	Environmental Details		
20. Identify key risks	Risks		
21. Identify key stakeholders	Stakeholders		
22. Identify implementation details	Implementation Details		
Work Processes			
23. Identify key work processes	Work Processes	Existing Process Project Managers Business Analysts Domain Experts	Interviews Document Analysis Application Analysis
Functional Aspects			
24. Identify key functional aspects	Functional Aspects	System Manuals Existing Process Project Managers Business Analysts Domain Experts	Interviews Document Analysis Application Analysis
Non-functional Aspects			
25. Identify key non-functional aspects	Non-functional Aspects	System Manuals Existing Process Project Managers, Business Analysts, Domain Experts	Interviews Document Analysis Application Analysis

Presentation of the Scoping phase consists of documenting the results of the workshop sessions, checking these informally for quality, and then distributing the resultant Vision & Scope document (Wiegers 2003) to the workshop participants for review and feedback, either as a group in a walkthrough, or individually as an inspection. Ideally the Project Sponsors should approve the updated version of the Vision & Scope before the elicitation project proceeds to the High-level phase.

The High-level Phase

In addition to studying all of the available high-level internal documentation sources relevant to the project, such as organisation charts and departmental reports, the Preparation stage of the High-level phase also requires the participating analyst to observe and take summary notes on the existing work processes and system operations relevant to the target system and the established scope. This enables the analyst to achieve a basic understanding of the business processes, and therefore guide the subsequent workshop in a more informed way with that knowledge.

The Performance of the High-level workshop (15 method components as can be seen in Table 4.5.6 below) involves firstly the tasks of reviewing and refining the information elicited from the Scoping workshop. In addition to the project team, High-level workshops also typically include Domain Experts, Middle Management, and Key User Representatives as sources. During this stage, the system environment is examined in detail, and the main work processes, features, and capabilities of the target system are identified and described. This can be performed using a variety of techniques including Questionnaires, Domain Analysis, and Viewpoints

Table 4.5.6: High-level Performance Stage Method Components

Task	*Info Type*	*Source*	*Technique*
Business Context			
26. Refine project information	Project Information	Project Managers Business Analysts Domain Experts	Interviews
27. Refine deliverable information	Deliverable Information		
Application Setting			
28. Refine system information	System Information	Project Managers Business Analysts Domain Experts	Interviews Questionnaires
29. Refine system goals	Goals		
30. Describe system assumptions	Assumptions		
31. Describe system constraints	Constraints		
32. Describe system environment	Environmental Details		
33. Describe system opportunities	Opportunities		
34. Describe system challenges	Challenges		
35. Describe risks	Risks		
36. Describe stakeholders	Stakeholders		
37. Describe implementation details	Implementation Details		

Work Processes			
38. Describe work processes	Work Processes	Exiting Processes System Manual Project Managers Business Analysts Domain Experts	Interviews Document Analysis Application Analysis Domain Analysis Task Analysis Use Cases Scenarios
Functional Aspects			
39. Describe features and capabilities	Functional Aspects	Existing Process System Manuals Forms and Reports Project Managers Business Analysts Domain Experts	Interviews Document Analysis Application Analysis Domain Analysis Task Analysis Use Cases Scenarios
Non-functional Aspects			
40. Describe non-functional aspects	Non-functional Aspects	Existing Process System Manual Project Managers Business Analysts Domain Experts	Interviews Document Analysis Application Analysis Goals Questionnaires

Like the previous phase, the High-level Presentation stage consists of documenting the results of the workshop sessions, quality checking, and then distribution and review, except this time in the format of a Concept of Operations (ConOps) document (IEEE 1998b). Approval of the High-Level document, as with the Scoping and Detailed documents, may require several iterations of reviews and updates before approval is attained, depending on the effectiveness of the workshops, commitment of the stakeholders, and the complexity of the project.

The Detailed Phase

A major task of the Preparation stage for the Detailed phase is not only to review detailed internal documentation sources, such as work instructions and system manuals, but the analyst is also required to observe and take detailed notes on the existing work processes and system operations identified in the High-Level phase. As the participants for the Detailed workshops will typically include Supervisors, End

Users, and Developers, it is important that the analyst is very familiar with the specific activities that must be supported by the target system.

During the Performance stage (18 method components as can be seen in Table 4.5.7 below), it is necessary for participating stakeholders to review the results from the Scoping workshop in order to understand the objectives and constraints of the current project and target system. The work processes, and functional and non-functional aspects identified in the High-level workshop are then examined in detail with the relevant project stakeholders and system users. Each work process is decomposed into individual steps with exceptions and extensions using for example Use Cases or Scenarios. Likewise, each functional and non-functional aspect is further decomposed into individual functional and non-functional requirements, once again using a combination of requirements elicitation techniques such as Goal decomposition and Viewpoint definition for example.

Table 4.5.7: Detailed Performance Stage Method Components

Task	*Info Type*	*Source*	*Technique*
Business Context			
41. Review project information	Project Information	Exiting Processes System Manual Project Managers Business Analysts Domain Experts Supervisors Key Users	Interviews Brainstorming
42. Review deliverable information	Deliverable Information		
Application Setting			
43. Review system information	System Information	Exiting Processes System Manual Project Managers Business Analysts Domain Experts Supervisors Key Users	Interviews Brainstorming Questionnaires
44. Review system goals	Goals		
45. Review system assumptions	Assumptions		
46. Review system constraints	Constraints		
47. Refine system environment	Environmental Details		
48. Refine system opportunities	Opportunities		
49. Refine system challenges	Challenges		
50. Refine risks	Risks		
51. Refine stakeholders	Stakeholders		
52. Refine implementation details	Implementation Details		
Work Processes			
53. Refine work processes 54. Describe work process steps	Work Processes	Exiting Processes System Manual	Interviews Document Analysis

		Project Managers Business Analysts Domain Experts Supervisors Key Users	Application Analysis Domain Analysis Task Analysis Use Cases Scenarios
Functional Aspects			
55. Refine key functional aspects 56.Describe functional requirements	Functional Aspects	Existing Process System Manuals Forms and Reports Project Managers Business Analysts Domain Experts Supervisors Key Users Developers	Interviews Document Analysis Application Analysis Goals Scenarios Viewpoints Questionnaires
Non-functional Aspects			
57. Refine key non-functional aspects 58. Describe non-functional requirements	Non-functional Aspects	Existing Process System Manual Project Managers Business Analysts Domain Experts Supervisors Key Users Developers	Interviews Document Analysis Application Analysis Goals Questionnaires

The same process is followed once again for the Presentation stage of the Detailed phase, however the format is that of a full System Requirements Specification document (IEEE 1998a). Given that this document is substantially more comprehensive than the previous documents, and involves most of the participating stakeholders, finalization of this document and its subsequent approval can often take considerably more iterations and time. This is particularly the case when the document is to be used as part of a contractual agreement between a customer and supplier.

4.6 Discussion

It is important to remember that when developing a situational approach for an activity like requirements elicitation, a number of delicate balancing acts naturally take place. One of these is between the flexibility and the rigor within the approach. Another is the risk of being too specific, and hence limiting the applicability of the approach to only a small number of situations, against the risk of being too abstract, and therefore reducing the ability of the approach in providing the necessary support. Furthermore, requirements elicitation in particular is not a stand-alone process, but interrelated and interleaved with other development activities such as system design. Consequently, it is advantageous for any requirements elicitation approach to also be applicable to different software development models and methodologies.

As a result, we have endeavoured to provide the analyst with several ways of customizing the approach. This not only includes the ability to engineer situation methods for requirements elicitation from scratch, but also the capability to tailor custom and ready-made methods dynamically throughout the process. This is commonly referred to as ‘opportunistic planning’ (Yeaple 1992), where original plans are incrementally replanned throughout a process, based on new information and changing conditions. The approach is further supported by a process framework, which the analyst can adopt completely, partially, or disregard altogether. It is therefore envisaged that the approach, or the selected parts of it, will often be used as a prescribed method, a baseline, or an outline for requirements elicitation in real-world system development projects, which can be further refined and improved over time.

Within the approach, the actual execution of the requirements elicitation project is mainly task-driven, in order to optimise the process through structure and sequence. However construction of the method is largely info type-driven for the sake of completeness, since it is arguably more important that all the required information is elicited, rather than all the planned tasks are performed. Rather than causing a conflict, these task-driven and info type-driven situations actually ensure that the overall approach is both effective (i.e. all the required info types are included and

elicited), and efficient (i.e. tasks are performed productively and systematically). Although the Tasks, Info Types, Sources, and Techniques of the workshops in the three recommended phases are significantly different, the entire activity is closely integrated and concentrated on the common objective of producing a complete, correct, consistent, and clear documented set of requirements for the target system with supporting information.

Throughout the approach we have purposely endeavoured to keep the process of engineering the method, and the process of requirements elicitation itself, as lightweight as possible. This is in direct response to our objective of providing support for novice analysts, and projects without a specifically prescribed software development process. Wherever possible we have attempted to conform to, or at least not conflict with recognized ME practices (e.g. (Henderson-Sellers 2002), (Ralyté, J., Deneckère & Rolland 2003), and (Brinkkemper, Saeki & Harmsen 1999)), in order to guarantee a level of consistency. Although our focus has been on the early stages of requirements elicitation, we have been careful to ensure that the fundamental ME concepts used can be adapted and applied to other software development activities, and range of process models, that acknowledge the iterative and incremental nature of requirements elicitation.

4.7 Chapter Summary

The lightweight approach for requirements elicitation presented in this chapter provides a number of potential benefits over existing ones. It is both extensible and flexible in that it provides guidelines for each step of the approach, and the ability to engineer and tailor situational methods based on specific project characteristics. Using the approach to develop a situation method does not require significant expertise or substantial experience, and is therefore particularly suited to novice analysts. In addition, the approach is especially useful in projects lacking a defined software development process because it provides high-level guidance, and offers the necessary framework to ensure an efficient process, and effective results. Furthermore, it is not dependent upon the utilization of any other systems development process, and can therefore be used either independently or as part of a larger methodology.

The operation of the approach takes advantage of both collaborative elicitation by being workshop centric, and the combination of multiple techniques in support of and integrated within the requirements elicitation workshop environment. As part of the approach, we have introduced a number of new and novel concepts including that of a 'method component', representing a task based method building block, and 'ready-made' methods that provide the analyst with a pre-constructed situational requirements elicitation process for the specific project at hand. In fact, as far as we know, this is the first attempt to directly apply a situational method engineering approach specifically to requirements elicitation for software development projects. Tailoring of the resultant methods can be performed throughout the process, and even during performance of the requirements elicitation workshops themselves.

In the next chapter we will present the MUSTER tool, a CASE/CAME Group Support System (GSS), which embodies and enhances the OUTSET approach. The empirical evaluations of the OUTSET approach and MUSTER tool (Chapter 5) combination by way of a case study, case study experiment, and formal experiment, will then be presented in Chapter 6.

CHAPTER 5: The MUSTER Tool

5.1 Chapter Overview

In the previous chapter we described OUTSET, a flexible yet systematic approach to the early stages of requirements elicitation in software development, based on guided collaborative workshops, and the construction of a lightweight situation method, within a general process framework.

In this chapter we will give details of the MUSTER tool, which embodies and enhances the OUTSET approach, and is based on Computer Aided Software Engineering, Computer Aided Method Engineering, and Group Support Systems (Section 5.2). We will describe the development of the tool and its features (Section 5.3), and the plug-in architecture which provides the mechanism to provide intelligent support (Section 5.4). We will then present an application of the MUSTER tool for a real world project (Section 5.5). This will be followed by a discussion (Section 5.6) and summary (Section 5.7) of the entire chapter.

The purpose of this chapter is therefore to present a tool to make OUTSET more useable and useful to practitioners, as well as automating many of the manual tasks required by the approach (**Research Goal 3**). We also aim to offer researchers an example of how group support systems can be applied to very practical activities and situations in the software development process in order to improve both the process and the results (*Research Question 8*). The overriding intention of the tool and the underlying OUTSET approach, however, is to improve the effectiveness and efficiency of the requirements elicitation process for the development of software systems, whilst at the same time addressing some of the common issues and challenges often encountered in practice.

5.2 Background

Computer Aided Software Engineering (CASE) tools support one or more techniques within a software development method (Jarzabek & Huang 1998). These tools are attractive to use during activities such as design, coding, testing, and validation, because of their potential to provide substantial gains in quality, productivity, management, and communication (Hoffer, George & Valacich 2002). Furthermore, CASE tools have been found to be efficient in both research and practice for recording, retrieving, and manipulating system specifications (Pohl et al. 1994), partly by automating some aspects of the system development.

Computer Aided Method Engineering (CAME) tools support the construction and management of adaptable methods (Saeki, Tsuchida & Nishiue 2000). These tools are useful in automating part of the process of engineering a method, by which to conduct one or more of the various system development activities, by reusing parts of existing methods (Saeki 2003). In addition, CAME tools have shown to be successful in providing the appropriate amount of process guidance based on the specific needs of software development problems and projects (Dahanayake 1998).

A common criticism of CASE tools is that they do not provide appropriate supporting guidance for the development process (Pohl et al. 1994), which can be directly addressed by the integration of a CAME tool. This would result in a process-based environment whereby the users can select, create, and modify method components for specific system development activities, in addition to performing the required system development tasks. The Phedias (Wang & Loucopoulos 1995) environment, referred to as a "CASE shell", was an early attempt at producing a combined CASE and CAME tool, relevant and with similar goals to our own research project. This tool enabled a method to be modelled at a Meta-level (i.e. a CAME tool), and corresponding CASE tools designed, developed, and integrated within this model and environment in order to provide support for the various activities (i.e. also a Meta-CASE tool (Alderson 1991)). As a precursor to our own attempt, Phedias is of particular interest because it was specifically targeted towards the development of methods and models of non-functional requirements for software engineering.

Group support systems (GSS) (Nunamaker, Briggs & Mittleman 1996), or groupware, on the other hand, when used within the context of development projects, typically takes the form of a software-based tool focused on supporting communication, coordination, and collaboration within a team working towards common goals, on interconnected workstations, in shared workspaces (Ellis, Gibbs & Rein 1991). The use of a GSS for the purposes of our research is particularly appropriate because of the number of key functions often provided by groupware applications that correspond directly to many of the tasks involved in requirements elicitation. These include activities such as information sharing, document authoring, knowledge management, and providing a suitable framework for stakeholder interaction. Furthermore, Group Support Systems have been found to be highly successful in improving group meeting productivity, and outcomes in real world settings (Hickey, Dean & Nunamaker 1999), as well as enabling larger groups to collaborate faster, particularly when matched with a specific requirements elicitation process (Hannola, Elfvengren & Tuominen 2005).

Subsequently, the idea of combining a GSS with requirements elicitation has been relatively popular. In fact Hickey, Dean, and Nunamaker state that the challenges of gathering accurate requirements, the inefficiencies of user interviews, and the difficulty of achieving effective group meetings, were early driving forces for Group Support Systems research (Hickey, Dean & Nunamaker 1999). As a result, there has been significant attention in research directed towards integrating groupware and requirements elicitation (den Hengst, van de Kar & Appelman 2004; Tuunanen 2003; Venable & Travis 1999), and of particular note are tools such as GroupSystems (Hickey, Dean & Nunamaker 1999), which has been used to collaboratively define scenarios, AMORE (Wood, D. P., Christel & Stevens 1994), which utilizes advanced multimedia technology, and TeamWave (Herela & Greenberg 1998), which specifically addressed distributed software development.

In (Liou & Chen 1993), a Group Support System (GSS), the Joint Application Development (JAD) method, and Computer-Aided Software Engineering (CASE) tools, were integrated to support the requirements specification process. In this work it was identified that support for requirements elicitation, and specifically collaborative

requirements elicitation meetings, was a major function missing from CASE research and products. It was also acknowledged that in order for a GSS to be successful in supporting requirements elicitation, it must also be supported with an appropriate methodology for its use, however no such specific process guidance was given. Therefore, the additional integration of a CAME tool, as proposed by our research, would enable not only the development and utilization of a contextual and dynamic method, but also the integration of different techniques to support the overall requirements elicitation process.

As a result, the name ‘MUSTER’ was chosen for our combined CASE / CAME / GSS tool, because with it we aim to bring together or ‘muster’ the different info types, tasks, sources, and techniques of requirements elicitation for software development, into an integrated process within a workshop environment. The aim of MUSTER is therefore to implement the OUTSET approach, using it as the underlying knowledge model and process framework, offering both control and guidance, in what is referred to in (Pohl et al. 1994) as a “process-aware CASE tool”. However, unlike most CASE tools, ours is intended to be used by groups rather than individuals, by applying the additional principles of groupware applications and workshop facilitation.

5.3 Development of the Tool

There were a number of important constraints in the development of the MUSTER tool, which had significant impact on its design and construction. Firstly, the system had to be developed by the researcher, and with a timeframe acceptable to the overall schedule of the project. Because there was no budget allocated to the project, it was also necessary for the system to be developed using only available and free technologies. Naturally, the system needed to implement the OUTSET approach, and support interactive and incremental requirements elicitation workshops. Furthermore, it was decided that the system should be as platform independent as possible, and thereby be able to run on most standard computer hardware platforms and operating systems.

From the work described in the previous three chapters, a first prototype of the MUSTER tool was constructed, with a preliminary set of standard features, and a detailed list of requirements elicitation related tasks. This prototype was tested and evaluated at a relatively high-level by numerous people both familiar and unfamiliar with the larger research project, including the research supervisors and fellow researchers. Although the prototype was found to be structured and extensive, it was essentially static with only limited process flexibility. Based on this feedback, a second prototype was developed with considerably more focus on the functionality required to make the tool less constrictive, more dynamic, and offer appropriate situational support. This resulted in several changes to better support the underlying approach, and to provide a suitable foundation for the planned evaluations. The details of this final prototype are described in the following subsections.

5.3.1 High-level Requirements

The detailed list of the requirements used for the development of the tool (as can be seen in Appendix E) was based on the results of the literature review (Chapter 2), the survey of practice (Chapter 3), and the OUTSET approach (Chapter 4), in addition to the “wish list” of seventy requirements for Requirements Engineering techniques

proposed by Macaulay in (Macaulay, L. 1996). At a high-level, and in accordance with the goals of the research, the tool should 1) improve the process of requirements elicitation in terms of the time and effort required, 2) directly address some of the common issues and challenges often encountered during requirements elicitation in practice, and 3) provide a suitable framework for the process of requirements elicitation by providing the necessary support for novices analysts conducting collaborative workshops when working in software development projects. It is therefore important to note that the principle focus of the tool is to improve the process of requirements elicitation, rather than improve the quality of the results.

The main functional areas required within the tool were established as 1) visualization, navigation, and administration through the elicitation process, 2) externalisation, representation and organisation of the elicited information, 3) process guidance, 4) cognitive support, 5) task automation, 6) interaction assistance for the participants, and 7) education of the users on requirements elicitation, primarily by osmosis. Although the primary usage of the tool is by an analyst to facilitate a workshop, it was determined that the system should be able to be used not only within a group setting, but also during a traditional one-on-one interview by the analyst with a stakeholder, as well as offline and independently by both the participating analyst and stakeholders. Although this requirement did not affect the design of the tool in any major way, it did provide the tool with an extra and potentially useful element of flexibility.

5.3.2 Architecture and Technologies

The application of artificial intelligence (AI) during requirements elicitation offers the potential to provide the type of help a novice analyst might receive from being mentored by an expert, and stakeholders with the kind of advice and guidance offered by a specialist workshop facilitator. This idea is supported in (Scott & Cook 2003), where a classical blackboard system with autonomous agents based on a knowledge repository is suggested in order to achieve such goals. Because Requirements Engineering, and especially elicitation, is essentially a cognitive activity, AI presents an appropriate opportunity to address this activity by providing situational cognitive

support to both the analyst and stakeholders (Zeroual 1991). This is confirmed by (Maiden, N. A. M. & Sutcliffe 1993) which states that "requirements engineering is complex, error-prone, and in need of intelligent tool support to assist the capture, modelling and validation of requirements".

Subsequently, two basic architectural orientations were identified as being potentially suitable for the development of the MUSTER tool and its intelligent components, being 1) a Multi-agent system such as JACK and JADE, or 2) an Expert system using Lisp or Prolog for example. It was also determined however that the concept of 'intelligent plug-ins' was not only similar to that of having multiple agents work cooperatively, but was also consistent with the operation of a partitioned expert system. Furthermore, the use of a plug-in architecture would provide many of the advantages of both Multi-agent and expert systems (e.g. the ability to use artificial intelligence), and at the same time enable a much wider choice in the selection of implementation technologies.

Subsequently, and after a thorough evaluation of available technologies, it was decided that the tool would be an online browser-based application, and the sever-side components would be based on the LAMP platform (Linux operating system, Apache web server, MySQL database system, PHP scripting language) with HTML (Hyper Text Markup Language), JavaScript, VBScript, and CSS (Cascading Style Sheets) incorporated where necessary. The advantages of the specific technologies chosen include the fact that they are easy to learn, use, and therefore maintain, and are entirely open source and completely free of charge, with extensive Internet based support networks. Furthermore, the amount of time required to produce a working prototype of the basic functionally required using PHP with MySQL was anticipated to be less when compared to the other option of a Java with XML based system. Because of the underlying environment, the tool would also be portable, scalable, and most importantly, flexible with respect to the integration of other technologies, tools, and components, necessary for a successful plug-in architecture.

5.3.3 Data Repository

The foundation of the tool is a central **Data Repository** (DR), which enables the storage and retrieval of large amounts of requirements, and requirements-related data, elicited from the stakeholders during the workshops, as well as the components of the OUTSET approach, and configuration information about the tool, projects, and users.

5.3.4 User Interface

The **User Interface** (UI) of the tool provides the ability to navigate through the required tasks, whilst interacting with other components of the system. As can be seen in Figure 5.3.1 below, the 'Home' screen displayed after a successful login has three major areas, as described in detail later, being 1) the Task List, 2) the Main Window, and 3) the Advice Panel. In developing the UI, a number of recognised Internet resources were used, including (Rolston 2005), to ensure that the overall look and feel of the tool was both simple and consistent.

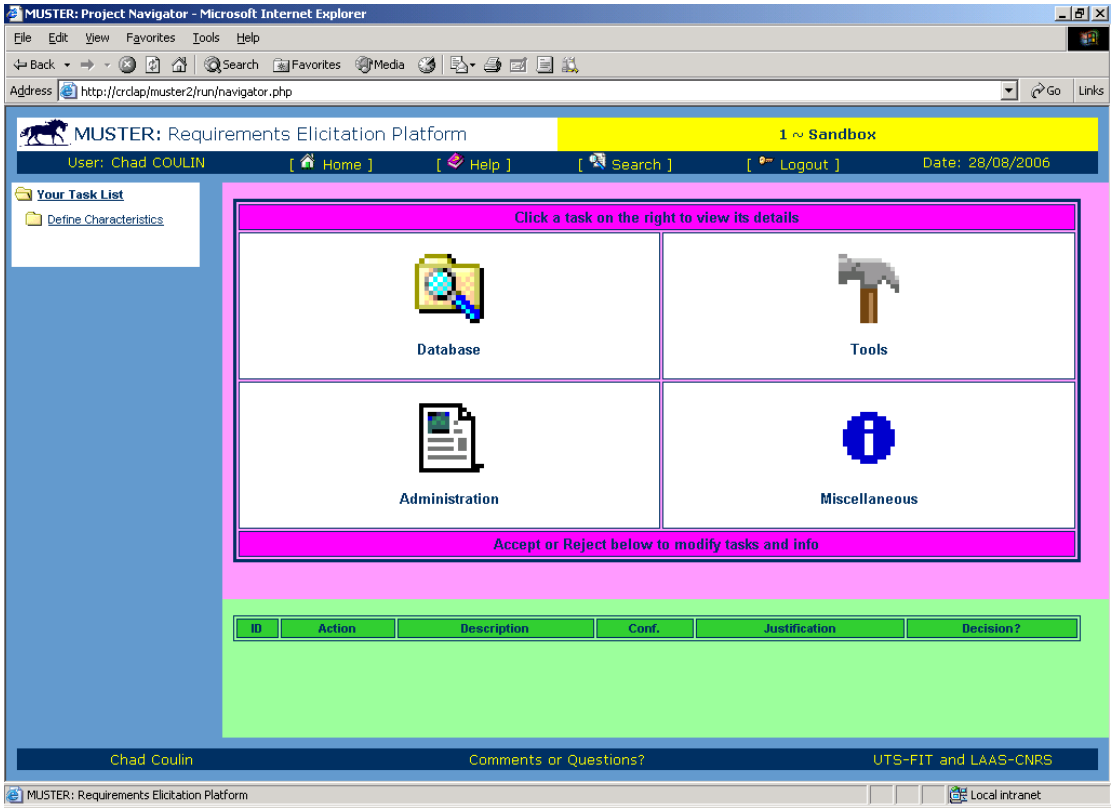

Figure 5.3.1: The MUSTER tool 'Home' screen

On the top row of the screen **Header**, the MUSTER logo and motto is displayed (white background) along with the name of the project currently logged into (yellow background). On the far left hand side of the bottom row of the Header, the name of the current user is displayed, and at the far right hand side of the bottom row of the Header, the current date is displayed. In addition, the bottom row of the Header contains the following functionality:

- **[Home]**: returns the user back to the 'Home' screen, as can been seen in Figure 5.3.1 above. The main window of this screen provides the user with access to the Database, Tools, Administration, and Miscellaneous menus described later.

- **[Help]**: contains all the available online documentation including the Getting Started Guide, the User Manual, the Workshop Guide, and the Technical Specifications for the tool.

- **[Search]**: enables the user to search the contents of the Data Repository (DR) using keywords or phrases. The user is able to enter a text string, and then select which areas of the DR the search should be conducted on.

- **[Logout]**: enables the user to log out of the system, thereby ending the session, and exiting the tool .

The screen **Footer** contains only three parts, the first of which on the far left hand side is a link to researcher's website. On the far right hand side are links to the UTS FIT and LAAS CNRS websites, and in the centre is a link for comments and questions about the tool that will automatically create a new email via the current users default email system, addressed to the researcher with the subject of "MUSTER".

The **Task List** (Figure 5.3.1, left hand side, blue background) provides a dynamically generated list of tasks for requirements elicitation process navigation and execution, which the workshop participants are recommended to perform during the various sessions. This task list is populated by the support plug-ins in accordance with the

underlying approach. Each task may be composed of several subtasks, and each task and subtask may have its own corresponding and specific Main Window.

The content of the **Main Window** (Figure 5.3.1, centre right hand side, purple background) is dependent on the task currently selected. There is no restriction on what should be displayed for each task, therefore the screen could be purely informational, or provide an editor for some related part of the Data Repository.

The **Advice Panel** (Figure 5.3.1, bottom right hand side, green background) presents the users with situational advice generated in real-time by the support plug-ins, based on the current state of the data in the repository (see Section 5.4 for more details)

5.3.5 Database Menu

The **Database** menu, accessible from the 'Home' screen, contains links to Data Repository (DR) maintenance screens for the available Info Types supported by the tool. By using these editors, which are based on the List-Detail-Post paradigm, users can select any Info Type in order to directly view, add, change, or delete specific entries within the DR for the current project.

5.3.6 Tools Menu

The **Tools** menu, also accessible from the 'Home' screen, contains the following features and functionalities:

[Glossary]: enables the user to maintain a project glossary by being able to add, change, delete, and view definitions of terms, acronyms, and abbreviations.

[Data Dictionary]: enables the user to maintain a record of data types within the project related to the system under investigation.

[References]: enables the user to maintain references to other material sources related to the project and/or the target system.

- **[Appendixes]**: enables the user to maintain a list of the required appendixes to the deliverables that will be generated as a result of the project.

- **[Issues]**: enables the user to record project related issues that arise during the workshops, as well as their status, who they are assigned to, and their resolution.

- **[Actions]**: enables the user to record actions that need to be performed during the project, as well as their status, who they are assigned to, and their resolution.

- **[Idea Hotpots]**: enables all users to record and maintain miscellaneous suggestions and proposals related to the project in order for other users to respond and comment on them anonymously at any time.

- **[Reports]**: provides a list of onscreen and exportable reports, enabling the user to produce deliverables from the information stored in the Data Repository (DR), for the purpose of reviews, walkthroughs, and inspections.

- **[Resources]**: provides various additional resources and material for the users, including templates, examples, and checklists, in order to further support the process of requirements elicitation and the workshop participants.

5.3.7 Administration Menu

The **Administration** menu, also accessible from the 'Home' screen by users with Administrator access (typically the participating analyst only), contains the following features and functionalities:

- **[Projects]**: allows the user to maintain projects in the system.

- **[Users]**: allows the user to maintain user accounts in the system.

- **[Sessions]**: allows the user to record details of the various requirements elicitation sessions performed during a project, including the start time, end time, participants, and location, for generate reports and performance metrics.

- **[Tasks]**: enables the user to view, add, change, and delete tasks in the dynamically generated Task List for each project and workshop.

- **[Plug-ins]**: enables the user to view, add, change, and delete information about the support plug-ins of the system, includes being able to install, enable and disable them.

- **[Configuration]**: enables the user to view information about the various configuration parameters of the MUSTER system, and change their values.

- **[Rules]**: enables the user to maintain rules and rule sets used by the Processor (see Processor in the Miscellaneous Menu below for more information). The system has default sets of rules however these can also be customized.

5.3.8 Miscellaneous Menu

The **Miscellaneous** menu, also accessible from the 'Home' screen, contains the following features and functionalities:

- **[Messages]**: This feature enables the users to record, view, and update messages in the system for other project members, but primarily the participating analysts, the project managers, and the MUSTER system administrators.

- **[Logs]**: This feature captures and records all events and actions performed by the users and the system. These include logins, logouts, as well as Add, Change, and Delete operations on data in the repository.

- **[About]**: This is purely an information screen that displays references and acknowledgements relating to the development of the MUSTER system, including contact details for support and feedback.

- **[Categorizor]**: This feature endeavours to categorize user entered pieces of miscellaneous textual information into their most appropriate Info Type, based on a small Artificial Neutral Network (ANN) that utilizes a number of key and common word lists. The concept behind this feature is that novice analysts are sometimes unsure as to how to categorize elicited information, especially with respect to goals versus requirements, and functional requirements versus non-functional requirements, for example.

- **[Processor]**: This feature enables the user at any time during the project to run one or more sets of rules over the information in the Data Repository to check for aspects of quality such as completeness, consistency, etc. For example, this feature could be used to ensure that all at least one actor has been assigned to each Use Case description elicited, or to check that each elicited feature has one or more individual functional requirements associated to it.

- **[Technique Selector]**: This feature, which utilizes a simple weighted values criteria approach, provides support for the user in selecting which technique to use for a task prescribed by the process guidance. The Technique Selector takes into account several factors, including the skill level of the participating analyst, the current project situation, and the specific characteristics of the task at hand.

- **[Ask REG]**: REG (Requirements Elicitation Guide) is a web-enabled pedagogic agent based on the famous and competition winning A.L.I.C.E. chat-bot engine, and AIML (Artificial Intelligence Markup Language). The intention is that REG acts as an interactive assistant by providing help for all MUSTER users, by responding to entered questions from a knowledge base of general information about requirements elicitation, and more specific information linked to a set of predefined topics and concepts.

5.4 Plug-ins for the Tool

MUSTER system plug-ins provide the situational process guidance and cognitive support for the users during the workshop sessions. The primary role of these plug-ins is to add, change, or delete tasks and subtasks in the project Task List, however they can also provide suggestions to the users dynamically, proactively, and reactively, such as introducing tasks, describing relevant background concepts, offering tips and tricks, as well as being able to directly manipulate data in the repository. As a result, plug-ins can provide the users with process guidance, decision support, and knowledge acquisition assistance. Plug-ins may provide generic support, or be based on a specific process or task, as well as a particular method (e.g. SADT, SSM, UML), technique (e.g. Scenarios, Viewpoints, Goals), or system type (e.g. Information, Embedded, Critical).

All advice generated by the installed and configured plug-ins appears in the Advice Panel of the screen, together with a justification and a confidence rating, generated internally by the specific plug-in responsible for that particular piece of advice. The advice provided by the plug-ins can be based on the characteristics of the individual project and workshop, as well as the information already elicited and stored in the data repository. Each piece of advice from the plug-ins presented in the Advice Panel may be rejected or accepted by the users, and an upper and lower threshold for the confidence rating is configured as a project characteristic to determine which advice is automatically accepted, and which is automatically rejected.

5.4.1 Plug-in Architecture

Regardless of the technologies used for its actual implementation, the architecture of a plug-in requires the following four components:

1. **Characteristics** – additional (non standard) situational characteristics which may be used by the conditions of the plug-in to determine which advice should be

offered and when. The ser is requested by the system to enter values for each of the new characteristics when the plug-in is run for the first time.

2. **Conditions** – represents the rules and logic of the plug-in, which is typical based on selected data from the repository, and the values entered for relevant characteristics. The conditions themselves can be implemented in almost any web-based technology, and can range from basic condition statements through to complex intelligent algorithms.

3. **Advice** – the specific situational support that may be offered by the plug-in. This is based on the triggering or results of the above mentioned conditions, and presented to the users for action

4. **Action** – the subsequent and prescribed result of accepting the offered advice. Each piece of advice offered by a plug-in will perform one or more operations in the system if accepted. These typically take the form of modifications to the task list, or manipulation of the data in the repository.

All the plug-ins are run automatically when any major event in the system occurs, which typically involves an Add, Change, or Delete operation on data in the repository. New advice generated by running the plug-ins is added directly to the bottom of the list presented in the Advice Panel with a status of ‘Open’. Once a piece of advice in the list has been either rejected or accepted, and the appropriate actions taken by the system, it is marked with a status of ‘Closed’, and removed from the Advice Panel list.

Advice can be presented, and appropriate actions performed, using the standard functions available in the Plug-in Function Library (PFL). Operations supported by the PFL include adding a Task or Subtask, adding an Info Type, checking if a particular piece of advice has already been given, and checking if a particular piece of advice was previously accepted or rejected. Depending on the configuration of the plug-in, a piece of advice may be presented only once for each project whether it is accepted or not, or it may be presented multiple times if not previously accepted for that particular project.

5.4.2 Creating Plug-ins

An overall plug-in architecture for the MUSTER system provides the ability for the tool to store and use the knowledge of both methodologists and requirements elicitation experts, within a comprehensive framework.

Plug-ins can be built using any single or combination of technologies, provided they are executable via the web server through a web page, including C++, VB, and Java. As a result, a wide audience is able to design and develop plug-ins for the MUSTER system, since no specific or proprietary technology-based expertise is required. The level of support, in terms of scope and complexity, is also not restricted, and at the higher end of the scale, almost any type of soft computing and machine learning method could be used as the basis for a plug-in, such as Bayesian Conditional Probability, Case Based Reasoning, and Artificial Neural Networks.

The process of installing a new plug-in is as simple as copying the relevant executable web page and associated files into the 'Plug-ins' directory of the MUSTER system. Using the Plug-in Maintenance Utility, details of the specific plug-in are then added to the system, including its status, and the name of the executable file in the directory. The final step is to enter values to the plug-in specific characteristics, which are installed the first time it is run. It is important to note that all plug-ins can be enabled or disabled at any time during a project using this same utility.

5.4.3 Plug-in Example

The example 'Select Info Types' plug-in, as summarized in Table 5.4.1 below, provides advice for the users on which Info Types should be elicited for each workshop, and to what level of detail they should be investigated. This plug-in uses a small Artificial Neural Network (ANN) (Young 2004) to determine the advice offered to the users, developed using a corpus of 15 example but real-world requirements documents from successful industrial projects and several widely accepted and used requirement document templates. The ANN was trained by running 10 examples

through the ANN 1000 times each, with the appropriate input values and corresponding output values, and the remaining 5 examples were then used to test the results of the trained ANN.

Table 5.4.1: Summary of the 'Select Info Types' plug-in

Name :	**Select Info Types**
Description :	Determines which Info Types should be elicited during the various requirements workshops
Technology :	Artificial Neural Network (ANN)
Characteristics :	Input Nodes for the ANN - 1) Project Size 2) Project Definition 3) Project Domain 4) Project Deliverable 5) Workshop Type 6) Cut-off Level
Conditions :	Output Nodes for the ANN - 1) Goals 2) Assumptions 3) Constraints 4) Environmental 5) Opportunities 6) Challenges 7) Risks 8) Stakeholders 9) Work Processes 10) Functional Aspects 11) Non-functional Aspects 12) Implementation
Advice :	Elicit each Info Type with a value above the cut-off level
Action :	Add Task for each Info Type suggested and accepted

In this case, the characteristics are used by this plug-in as input nodes for the ANN, and include the Project Size, Project Definition, Project Domain, Project Deliverable, and the Workshop Type. Each output node represents a potential Info Type, and the values generated by the ANN for a particular set of input node value, determines what Info Types are recommended for elicitation during the workshop. For each output node (Info Type) with a value above the cut-off level characteristic, an entry is displayed in the Advice Panel, which if accepted, will add a first-level task to the dynamic Task List stating the need to elicit that particular Info Type. Therefore, this plug-in determines which Info Types should be elicited (via the output node values) based on the specified characteristics (from the input node values).

5.4.4 Developed Plug-ins

This subsection contains descriptions of the initial set of plug-ins developed for the MUSTER tool. These plug-ins were developed both as a proof of concept for the designed and implemented architecture, and for the evaluation of the MUSTER tool.

5.4.4.1 New Requirements Elicitation Project

The ‘New Requirements Elicitation Project’ plug-in provides initial and core tasks and characteristics for new requirements elicitation projects. This plug-in primarily uses the ‘3Ds’ characteristics from the OUSTET approach to determine which tasks should be performed, and was designed to use a rules approach, based on a variety of sources from the literature including (Sommerville & Sawyer 1997) and (Robertson, S. & Robertson 1999).

5.4.4.2 Requirements Elicitation Workshop

The ‘Requirements Elicitation Workshop’ plug-in provides core tasks and additional characteristics for conducting requirements elicitation workshops. This plug-in also uses the ‘3Ds’ characteristics from the OUSTET approach to determine which tasks should be performed, and was designed to use a rules approach based on a variety of sources from the literature such as (Gottesdiener 2002).

5.4.4.3 Select Info Types

The 'Select Info Types' plug-in provides guidance on which Info Types should be elicited for the specific project and workshop, based on both project and workshop level characteristics. As described in Section 5.4.3 above, this plug-in was developing using an Artificial Neural Network (ANN) (Young 2004). A number of sources were used to design this plug-in including 15 example but real-world requirements documents from successful industrial projects, and several requirements specification templates including (IEEE 1998a), (Atlantic Systems Guild 2003), (IEEE 1998b), and (Wiegers 2003).

5.4.4.4 Select Goal Subtasks

The 'Select Goal Subtasks' plug-in provides guidance on which subtasks should be performed during the elicitation of both system and project goals. In addition to presenting information to the workshop participants about what a goal is, and how one should be stated, the plug-in also provides instructions on how to brainstorm and prioritise goals. This plug-in was based on separate sources for goal elicitation (Dardenne, van Lamsweerde & Fickas 1993), brainstorming, and prioritisation (Wiegers 2007).

5.4.4.5 Goal Investigation

The 'Goal Investigation' plug-in assists users to both decompose goals (by suggesting AND and OR relationships), and elaborate on goals (by proposing 'Why' and 'How' questions), in order to refine them in such a way as to elicit precise goals and related requirements. Several goal-based techniques for requirements elicitation were used as the basis for this plug-ins including (Yu 1997) and (Dardenne, van Lamsweerde & Fickas 1993).

5.4.4.6 Example BIS Constraints

The ‘Example BIS Constraints’ plug-in provides users with general and example constraints that are common or typical in software development projects for Business Information Systems. The intention of this plug-in is for the workshop participants to use the example constraints presented as the basis for the elicitation of similar constraints specific to the project at hand. The plug-in was designed based on the constraints listed in a relevant subset of 15 requirements documents from successful projects, and a variety of other sources, then implemented using a rules approach.

5.4.4.7 Use Case Questionnaire

The ‘Use Case Questionnaire’ plug-in proposes questions and provides suggestions to users on Use Cases that the system under investigation should support. This plug-in was implemented using Bayesian Conditional Probability (Meagher 2004), and uses Use Cases previously elicited and stored in the data repository, with their corresponding characteristic values, as the basis for questioning the workshop participants about the required Use Cases, and suggesting additional Use Cases to include.

5.4.4.8 IEEE Functional Headers

The ‘IEEE Functional Headers’ plug-in uses an Artificial Neural Network (ANN) (Young 2004) to determine the most appropriate way to group functional requirements (i.e. by mode, by user class, by object, by feature, by stimulus, or by functional hierarchy), based on the options presented in the IEEE Recommended Practice for Software Requirements Specifications (IEEE 1998a). Coded values for project characteristics such as the ‘3Ds’ are used as the input nodes of the ANN, with the weighted value of the output nodes representing the relative appropriateness for each of the available options for grouping the functional requirements.

5.4.4.9 Features Questionnaire

The 'Features Questionnaire' plug-in proposes questions and provides suggestions to users on features that the system under investigation should include. This plug-in was implemented using Bayesian Conditional Probability (Meagher 2004), and uses features previously elicited and stored in the data repository, with their corresponding characteristic values, as the basis for questioning the workshop participants about the required features, and suggesting additional features to include.

5.4.4.10 Feature Investigation Questionnaire

The 'Feature Investigation Questionnaire' plug-in dynamically generates a project specific questionnaire, to be used by the workshop participants for the elicitation of functional requirements. A list of high-level questions is created by the plug-in from a simple rule set developed from the researcher's experience, which can then be used to investigate and decompose each feature that has been elicited into specific functional requirements.

5.4.4.11 Non-functional Requirements

The 'Non-functional Requirements' plug-in provides users with additional support information for the elicitation of non-functional requirements. This includes a number of definitions and a list of typical non-functional requirements types gathered from a number of sources in the literature including (Sommerville 2001), with relevant examples.

5.4.4.12 Example BIS Non-functional Requirements

The 'Example BIS Non-functional Requirements' plug-in provides users with general and example non-functional requirements that are common or typical in software development projects for Business Information Systems. The intention of this plug-in is for the workshop participants to use the example non-functional requirements presented as the basis for the elicitation of similar non-functional requirements

specific to the project at hand. The plug-in was designed based on the non-functional requirements listed in a relevant subset of 15 requirements documents from successful projects, and a variety of other sources, then implemented using a rules approach.

5.5 MUSTER Tool in Action

The following section provides a basic walkthrough of the functionality and usage of the MUSTER system using a typical but simple project for the custom development of a small business information system. Although not all the features and utilities are demonstrated, this example does provide an overview of the general process of requirements elicitation using MUSTER within a workshop environment. In accordance with the process framework prescribed by the OUTSET approach, we have divided the following illustration of the system into three simple stages being 1) Preparation - setup of the new project in MUSTER, 2) Performance - running the workshops using MUSTER, and 3) Presentation - production of the requirements document from MUSTER, as described below.

5.5.1 Preparation

The first step in the preparation of a new MUSTER project is for the participating analyst to log into the maintenance utility of the system using an account with Administrator privileges. From the MUSTER Maintenance Utility menu, the analyst can then add a new Project and new Users to the system, as well as selecting the appropriate plug-ins to use. Each plug-in has a default status (either ‘Active’ or Disabled’), which provides a standard configuration of active and disabled plug-ins for new project, that can be then modified by the analyst. As we can see from Figure 5.5.1, only some of the installed and available plug-ins within the system have been enabled by the analyst for this particular project.

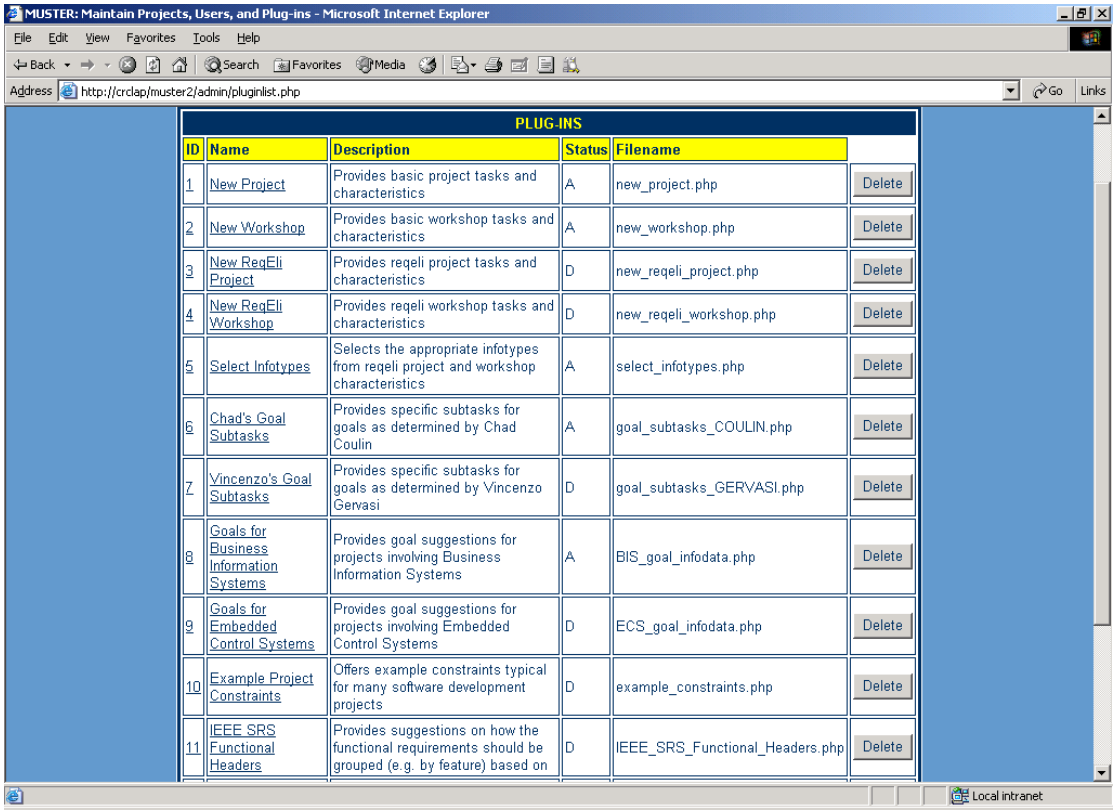

Figure 5.5.1: Plug-in maintenance utility of the MUSTER system

The analyst can then immediately log out of the MUSTER system, and log back into the newly created project in order to enter values for the situational project characteristics, required by the select plug-ins, as the first task of the new project (see Figure 5.5.2 below). The ‘Define Characteristics’ task has been added to the situational Advice Panel, and subsequently the dynamic Task List, by the ‘New Project’ plug-in. The characteristics for each of the enabled plug-ins have been added, and in some cases loaded with default values, automatically the first time they are run, which in this case was triggered by the event which added the ‘Define Characteristics’ task to the dynamic Task List.

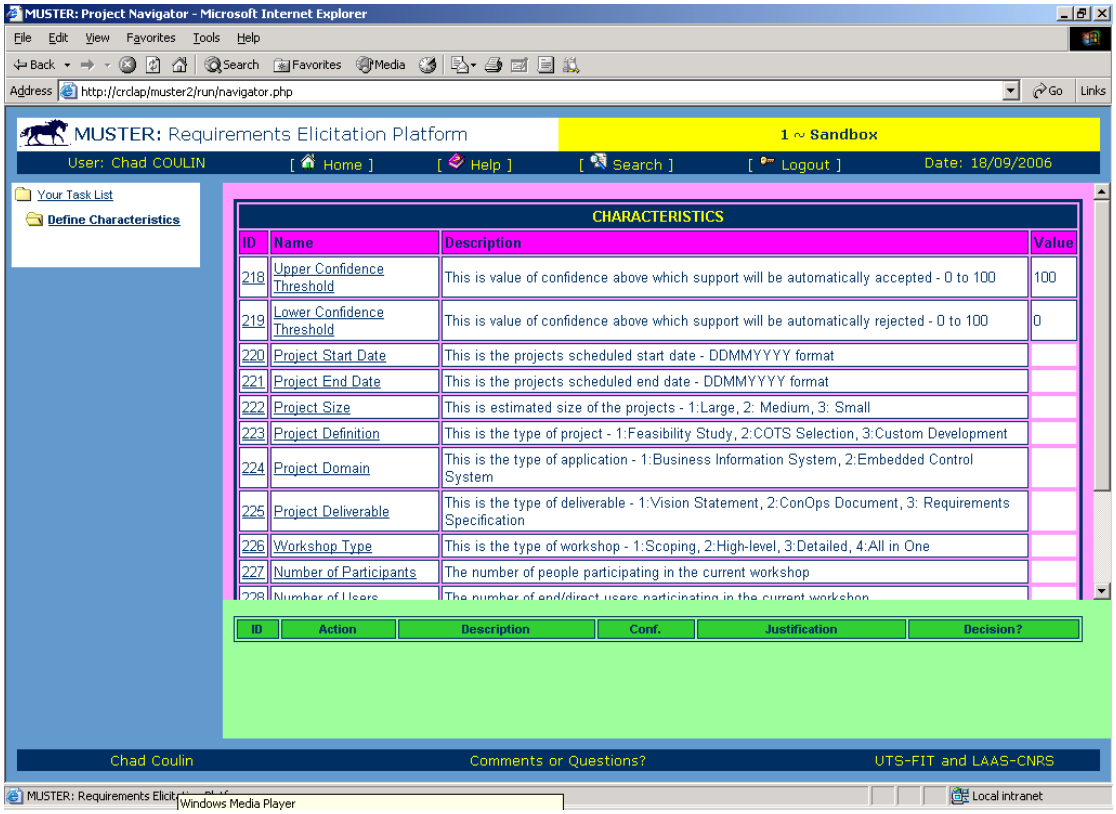

Figure 5.5.2: Situational project characteristics maintenance screen

For this example, the current project is small in size, and involves the production of a requirements document for the custom development of a business information system. Furthermore, the intention is to elicit as much of the information as possible in a simple combined workshop with multiple sessions.

5.5.2 Performance

As values for the characteristics are entered, tasks are added to the list, and data is maintained in the repository, the plug-ins are repeatedly triggered, and are therefore able to start and continue providing situational support to the users via the advice panel. From Figure 5.5.3 below, we can see that after only some of the characteristics values have been added, already the system via the plug-ins has provided several items of advice with respect to the core tasks that should be added to the dynamic list, including eliciting system goals, and eliciting project constraints, among others.

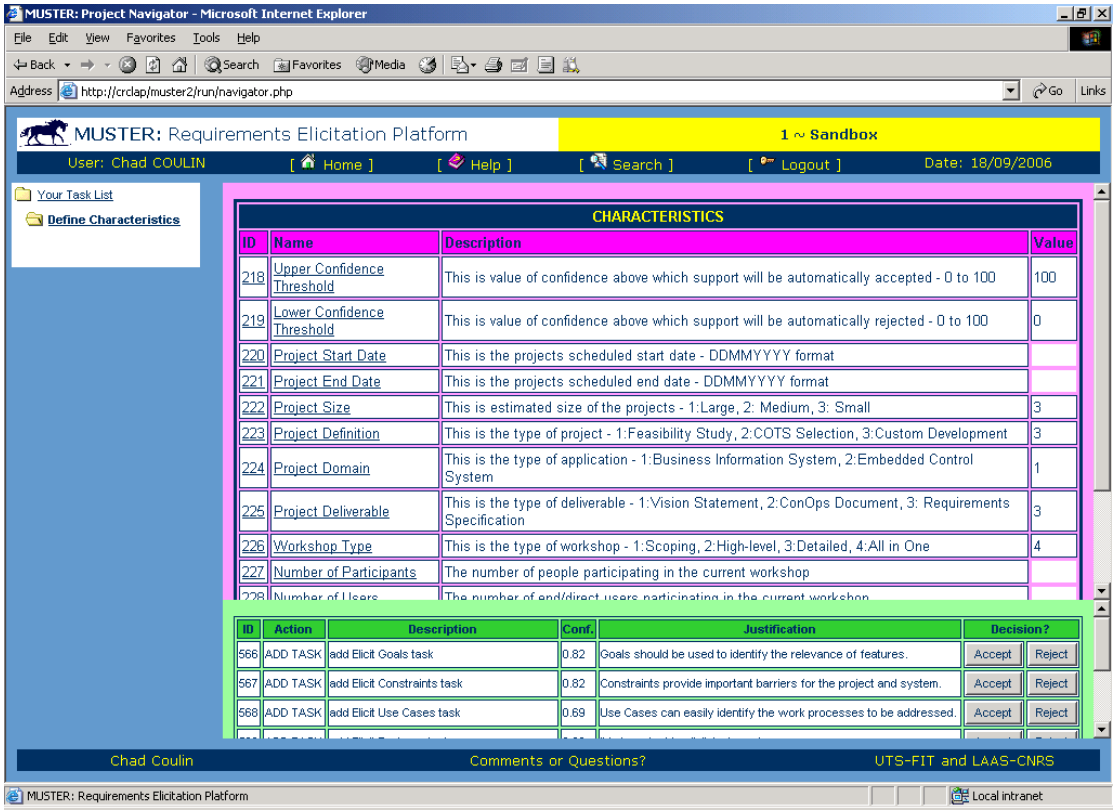

Figure 5.5.3: System screen shot after some characteristic values entered

Both the analyst and the stakeholders participating in the requirements elicitation workshop make their way through the task list, adding data to the repository for the recommended and accepted Info Types, through task specific instructional and maintenance screens. Task specific advice from the plug-ins, including the addition of subtasks, may in some cases only be offered if the dependent first-level task has been added, and once that particular first-level task has been started.

In a screen shot taken towards the end of the workshop (Figure 5.5.4), we can see that several more tasks and sub-tasks have been added to the list. Furthermore, the only item remaining in the advice panel relates to the presentation of the data in the repository, by way of quality check and export, in the format of the required deliverable type. In this case, the 'Presentation' task has been offered by a plug-in and added to the advice panel only after each of the information types has had at least one entry recorded for it in the MUSTER system.

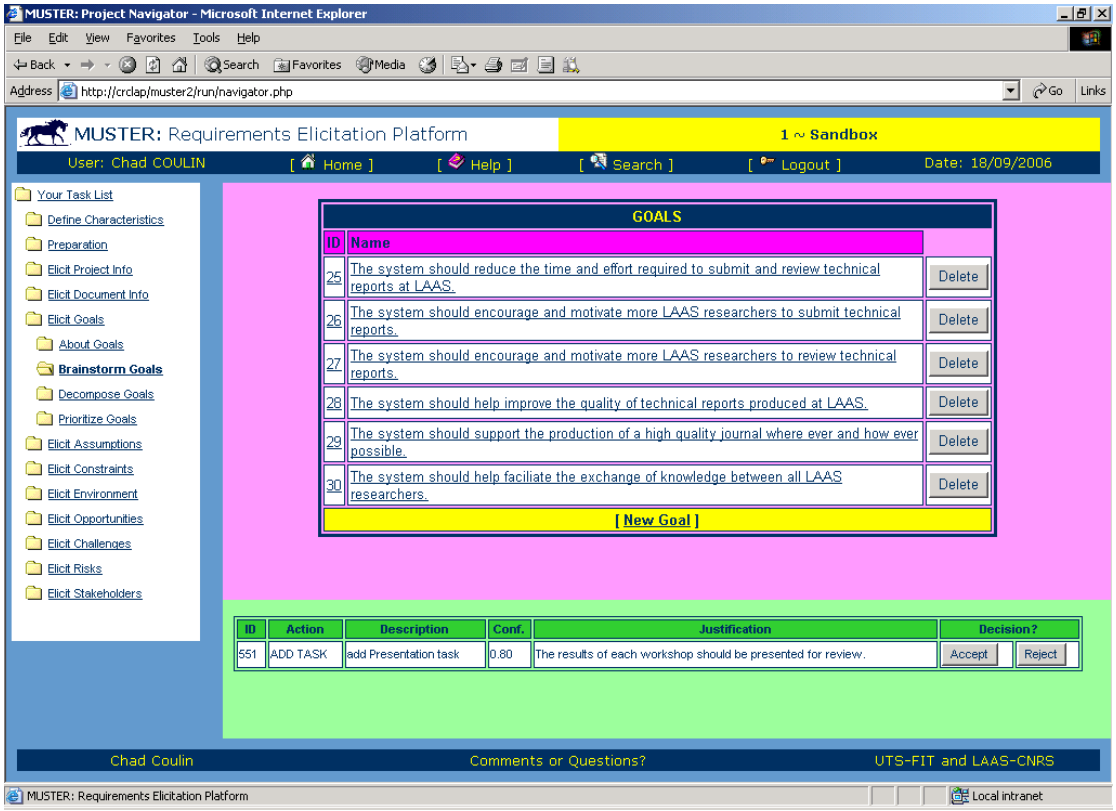

Figure 5.5.4: Screen shot of the MUSTER system near the end of the workshop

The workshop may come to a conclusion either 1) when the participants have run out of allocated and available time, or 2) when the participants can no longer think of additional relevant data to enter into the MUSTER system, and the plug-ins can no longer offer meaningful advice to them. At this point the project is ready for presentation as described below.

5.5.3 Presentation

Before exporting the results of the workshop out of the MUSTER system, the elicited information can be reviewed collaboratively, or individually, by the participants. In order for the necessary walkthroughs and inspections to take place, the data from the repository is formatted and exported using one of the available standard reports (see Figure 5.5.5 below).

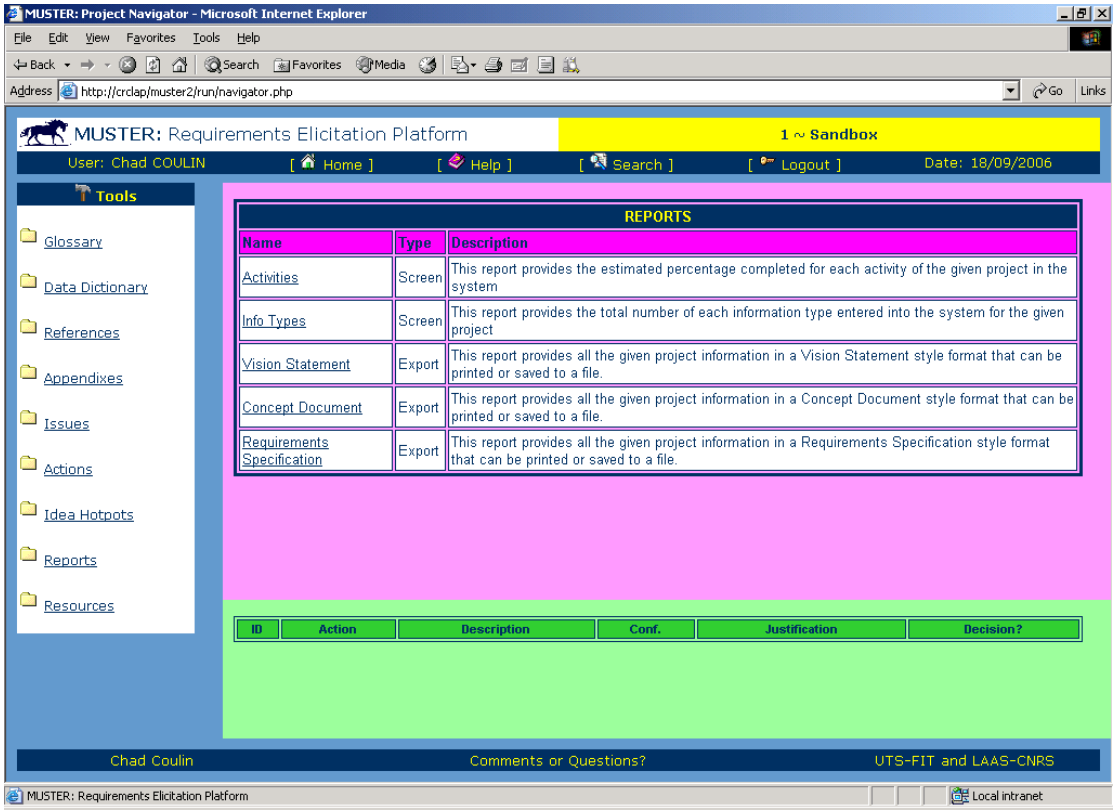

Figure 5.5.5: List of available standard reports in the MUSTER system

The resultant deliverable of this process, as produced by MUSTER, is then ready for feedback and approval by the appropriate workshop participants and other stakeholders.

5.6 Discussion

For the MUSTER system, a number of additional features were considered but not implemented, including Instant Messaging, Discussion Boards, Reuse Utility, and a Workflow Engine. The reasons for their exclusion were for the most part related to the potential benefit they would provide, compared to their relevance to the overall research goals of the project. Also considered useful but somewhat out of scope with respect to the focus and objectives of the tool, included the use of visual effects such as zooming user interfaces, mouse over magnification, and wheel or web data representations. In addition, several features such as the Online Help, Categorizor, Technique Selector, and Ask REG, were only partially developed despite their novelty, because of the effort required to implement and evaluate them to a level that would provide substantial enhancement to the research results.

The use of a simple and open plug-in architecture has given the tool a number of important advantages, including the ability to utilize different technologies, and offer different types of support to the users. However because these plug-ins can originate from many sources, and may be based on subjective and imprecise reference material (e.g. experience), the potential result of all the advice offered by the combined plug-ins is difficult to predict. Consequently the effectiveness and usefulness of the MUSTER system is heavily dependent on the quality of the advice offered by the plug-ins, and the way in which expert knowledge is presented within the workshop environment. As a result, it is not possible to claim that the tool is as good or better than having a participating requirements elicitation expert facilitate the workshops, but rather that the support offered will be of benefit to novice analysts in guiding the process of requirements elicitation

Furthermore, in addition to providing an implementation of the situational OUTSET approach, MUSTER addresses some of the issues often experienced in requirements elicitation practice. For example, the tool allows end users of the system to communicate openly, access project information, and be actively involved in both the elicitation process and the development of the target system requirements, thereby encouraging project ownership and stakeholder commitment. MUSTER also

overcomes a major limitation of many groupware applications (Alho & Sulonen 1998), in that it not only supports, but also actively encourages, the use of a dynamically generated process based on contextual factors. In order to achieve this, we have endeavoured to make the tool and its usage as flexible and configurable as possible, whilst still providing an appropriately structured and rigorous foundation for requirements elicitation, creative thinking, idea crystallization, and constructivist learning.

5.7 Chapter Summary

In a retrospective report on lessons learnt from ten years of Group Support Systems research (Nunamaker, Briggs & Mittleman 1996), it was determined that a GSS can significantly reduce time and effort, but does not replace leadership. Likewise, the MUSTER system is intended to provide practical benefits and support for the analyst and stakeholders during requirements elicitation, rather than replace the role of a workshop facilitator completely. In the same report, it was stated that Group Support Systems should include separate special purpose modules to permit flexible process design, which we have also endeavoured to satisfy through our use of an overriding plug-in architecture for the usage of the tool and generation of situational guidance.

In terms of being a virtual workbench for requirements elicitation, and as a special purpose CASE/CAME application, MUSTER provides a number of potential benefits over existing requirements elicitation tools. Through direct interaction with the analyst and stakeholders during the workshops, MUSTER removes the need for costly and time-consuming meeting transcription and report writing, whilst still being collaborative and combinational. The utilization of web-based technologies, and the integration of intelligent plug-ins, enables contextual guidance and support to be provided through an integrated situational approach and environment.

Potentially the most relevant and important aspect of the MUSTER system is that utilization of the tool does not necessarily require significant expertise or substantial experience, nor is it dependent upon the selection and implementation of any other process for the rest of the software development activities. As a result, we believe that MUSTER, like the OUTSET approach it is based on, is particularly suited to novice analysts and those projects lacking a defined requirements elicitation method. However, the acceptance and adoption of any GSS, CASE, and CAME tool is also heavily dependent on the ease of learning, and its appropriateness to the situation. It is these and other aspects related to the MUSTER tool and OUTSET approach that will be tested through evaluation in the following chapter.

CHAPTER 6: Empirical Evaluations

6.1 Chapter Overview

In the previous chapter we described the MUSTER tool, a computer aided software engineering (CASE) and computer aided method engineering (CAME) group support system (GSS), which embodied, enhanced, and extended the OUTSET approach. In this chapter we will present the evaluation framework for a specific instance of the OUTSET approach and MUSTER tool (Section 6.2), involving a case study (Section 6.3), a case study experiment (Section 6.4), and a formal experiment (Section 6.5). This will be followed by a discussion (Section 6.6) and summary of the entire chapter (Section 6.7).

The purpose of the evaluations described in this chapter was to provide empirical evidence as to the performance of both the OUTSET approach and MUSTER tool for requirements elicitation, in terms of improvements to efficiency (*Research Question 10*), effectiveness (*Research Question 11*), and useability (*Research Question 12*). We therefore directly address our **Research Goal 4** in this chapter by empirically evaluating the efficiency, effectiveness, and useability of both the approach and tool for the early stages of requirements elicitation, when performed by novice analysts in the absence of a defined methodology.

6.2 Evaluation Framework

The available literature on the empirical evaluation of software development approaches and tools is somewhat limited (Kitchenham, B. et al. 2006; Taylor & Urban 1994), however we have endeavoured to develop a structured and rigorous evaluation framework, as described later in this section, for both the approach and the tool with respect to the goals of the research and the available time and resources, based on the following review and critical analysis of the existing evaluation methods.

Case Studies are the investigation of real-life phenomenon in their natural context, and not the application of theory to hypothetical situations. Case study research in general is strongly associated with the disciplines of sociology, psychology, medicine, law, education, and business. The advantages of case studies (Kitchenham, B. A. 1996a) as a research method for software engineering include 1) they can be incorporated into the normal development activities, 2) they show the real effects of the approach/tool in real situations, and 3) they enable validation of the work by actual practitioners. The disadvantages of case studies (Kitchenham, B. A. 1996a) are 1) that with little or no replication they may give inaccurate results, 2) there is no guarantee that similar results will be found on other projects, and 3) there are few agreed standards/procedures for undertaking case studies in software engineering. In the past some have argued that case studies are extremely limited in providing conclusions that can be generalised, given they are based on a single set of circumstances. However over time this has regularly been rejected, stating instead that case studies are acceptable and can provide very meaningful information provided they are designed and performed with the appropriate degrees of rigor and structure (Hamel, Dufour & Fortin 1993).

Case studies represent an appropriate choice for our goals as they enable us to evaluate our solution in real-world organisations, which is vital given our research is based on addressing practical problems. According to the evaluation selection criteria in (Kitchenham, B. A. 1996b), our approach and tool is most suited to a case study because 1) the benefits are difficult to quantify, 2) the benefits are observable on a single project, and 3) the timescale for the evaluation is commensurate with the

elapsed time of a normal size project. Furthermore, a number of similar examples exist in the literature where the selected case study method has been successfully used for the evaluation of software engineering approaches and tools including (Cybulski 2002) for requirement specification reuse and refinement, and (Chatzoglou 1997) for social and organisational requirements elicitation. Subsequently we have used the guidelines of Yin (Yin 1994) as well as Kitchenham, Pfleeger, and others in (Kitchenham, B., Linkman & Law 1997; Kitchenham, B., Pickard & Pfleeger 1995; Kitchenham, B. A. 1996a; Kitchenham, B. A. et al. 2002) and more specifically (Kitchenham, B. A. & Pickard 1998a, 1998b), for designing, conducting, analysing, and presenting our case study research. The work of Yin is widely regarded as seminal, and is possibly the most cited reference for general case study research currently. The DESMET methodology developed by Kitchenham et al. (Kitchenham, B., Linkman & Law 1997) is also useful for our purposes as it supports the singular and specific evaluation of approach/tool combinations in software engineering.

Formal Experiments are where a number of groups use different approaches or tools to do the same task, under the same conditions and circumstances, for the purposes of comparison. Formal experiments are useful for investigating alternative approaches of performing self-standing tasks, i.e. those that can be isolated from the overall product-development process, and "are essential if you are looking for results that are broadly applicable across many types of projects and processes" (Kitchenham, B., Pickard & Pfleeger 1995). Experiments have the advantage of being more structured, and therefore repeatable under the same controlled conditions. But they also have the disadvantage of not necessarily reflecting real practice, where the conditions are rarely controlled or the same. In general the scope of formal experiments is typically smaller than case studies by necessity, as in the former it is important to have control over the variable factors. Experiments also have a stronger theoretical basis than case studies, and are therefore less susceptible to bias. Although case studies may be easier to plan than experiments, their results are harder to interpret and difficult to generalize (Kitchenham, B., Pickard & Pfleeger 1995).

As detailed in (Kitchenham, B. A. 1996b), the use of formal experiments is also appropriate for the aims of our research given that 1) the approach and tool are related to a single activity (i.e. requirements elicitation), 2) benefits are directly measurable

from the task output, 3) relatively small learning time is required, and 4) there is a desire to make context independent assessments. Numerous examples of experiments being used to evaluate approaches and tools for requirements elicitation exist in literature including (Hickey, Dean & Nunamaker 1999) where Hickey, Dean, and Nunamaker investigated collaborative scenario elicitation using a collaborative tool called GroupSystems. Another instance is (Lloyd, Rosson & Arthur 2002) where Lloyd, Rosson, and Arthur evaluated the effectiveness of elicitation techniques in distributed requirements engineering using a real-time virtual meeting support software called Centra Symposium and a second tool called MOOsburg for file sharing and asynchronous discussions. Similarly, Moore and Shipman (Moore & Shipman 2000) used experiments to test their Graphical Requirements Collector tool when compared with a traditional requirements elicitation questionnaire, much like the work of Gambhir (Gambhir 2001) who used facilitated and non-facilitated groups with and without document templates to compare the communication, comprehensiveness, and quality of the process and results in requirements elicitation.

Surveys typically involve a broad but basic review of multiple projects that have already performed across different or the same contexts, and are more appropriate for state-of-practice investigations such as those in (Hickey & Davis 2003b), (Hofmann & Lehner 2001), and (Chatzoglou 1997). Although the results produced from surveys are more easily generalised because it is possible to use proven statistic techniques for analysing the data, a survey alone was not an option for our research because the approach and tool we wished to evaluate were still in development and not readily available or used in practice.

Feature Analysis is another alternative evaluation method for software tools, which in essence uses a set of Yes/No questions about the features of comparison tools that are prioritized, ranked, and scored. Unfortunately, there were no other tools readily available to practitioners with a similar set of features to MUSTER that could be used for comparison. Although this method is relatively easy to implement, it is too simple for our needs, and does not address the main goals of our research. Furthermore, feature analysis is mainly limited to tools that perform a set of functions, rather than approaches that aim to improve a process.

Useability Tests may also have been used to evaluate our approach and tool however these alone would not enable us to evaluate the system sufficiently to address all the aims of the research project, such as the impact on the amount and relevancy of information elicited. In useability tests the individual participants are typically given a set of tasks to complete and asked to think aloud while performing them. Their comments are recorded, as are any problems encountered during the completion of the task, and the time taken to complete each task. This method by itself is once again not appropriate for our research given that we are primarily interested in examining group interaction with the system, rather than individual performance, with complete freedom of operation during the larger activity being essential for the evaluation.

Other potential methods included **Action Research**, **Ethnography**, and **Grounded Theory**. Action Research (Mills 2002), where both the researcher and the participant are collaboratively engaged in a study to solve real problems and achieve real goals, was not selected as there is still some debate over the validity of its usage for information technology based research. Ethnography (Fetterman 1997) was not an appropriate choice because the goals of the research did not require or desire the complete immersion of the researcher into the lives of the participants, and because the context of the phenomenon under investigation was not of primary importance. Since the problems and solutions initially identified in our research were drawn largely from the literature, Grounded Theory (Dey 1999) was also not relevant to the way we planned and conducted the work, as this method advocates the development of the theory continuously as part of the research process through iterations of data collection and analysis.

Consequently, a case study (Section 6.3), a case study / formal experiment combination (Section 6.4), and a formal experiment (Section 6.5) were planned and conducted with the goal of evaluating our research by validating the underlying OUTSET approach and its specific implementation in the MUSTER tool. This is in direct accordance with (Kitchenham, B. A. et al. 2002) which prescribes that it is important to first understand how the system works in an industrial setting before developing an abstract version for formal experiments. The three evaluations were deliberately complementary, in that they allowed us to evaluate the approach and tool 1) in a real-life project, then 2) with novice analysts in a real-life project, and then 3)

with novice analysts in a controlled environment comparable with the first two evaluations. In all of these evaluations, the three main aspects of the combined approach and tool measured were *efficiency*, *effectiveness*, and *useability* as described below.

Efficiency, which refers to the resources expended to complete a task (Bevan 2006), can be measured using indicators such as task completion times, and the quantity of results (Frøkjær, Hertzum & Hornbæk 2000). As a result, and with respect to the efficiency of our approach and tool combination, the measurement we have chosen is the amount of information elicited, meaning the overall number of individual and self-contained pieces of information elicited (e.g. all goals, features, and requirements). This measurement was selected because it relates directly to the quantity of the information elicited during the requirements elicitation activity for the effort expended. Although the expended effort compared with the overall amount of information elicited is only one factor which could contribute to a measurement of efficiency (others possibilities include the total cost of the activity and the resources used), it is arguably the most evident, and is therefore relatively easy to identify, count, and compare.

Effectiveness, which refers to the accuracy and completeness of a task (Bevan 2006), can be measured using indicators such as the quality of the results, and the number of errors made (Frøkjær, Hertzum & Hornbæk 2000). As a result, and with respect to the effectiveness of our approach and tool combination, the measurement we have chosen is the quality of information elicited, meaning the number of individual and self-contained pieces of relevant information elicited (e.g. only validated goals, features, and requirements). This measurement was selected because it relates directly to the quality of the information elicited during the requirements elicitation activity for the effort expended. Although any determination of what is and is not relevant information for the effort expended will be subjective to an extent, the cross validation process described later in Section 6.5.1 goes some way towards ensuring that the subjectivity for this determination is at least consistent across the evaluations.

The International Organization for Standardization (ISO) defines **useability** as “the extent to which a product can be used by specified users to achieve specified goals

with effectiveness, efficiency, and satisfaction in a specified context of use" (ISO 1998). As a result, we developed a standard feedback questionnaire to be used for all three of the evaluations to be described in the following sections, which necessarily covers the three different aspects of effectiveness, efficiency, and satisfaction (i.e. the users' comfort with, and attitudes towards, the use of the tool (Frøkjær, Hertzum & Hornbæk 2000)). Using this standard questionnaire not only enables us to compare the results across the three evaluations, but also across the different tools used in the formal experiment (Bevan 2006). Our useability questionnaire was based primarily on the SUMI (Kirakowski & Corbett 1993) and SUS (Brooke 1996) useability questionnaires as these are both widely regarded and commonly used in research and practice (e.g. (Sauro & Kindlund 2005)).

6.3 LAAS Case Study

The first step in the evaluation of the OUTSET approach and MUSTER tool was an exploratory case study project within LAAS, with the goal of evaluating the performance of our approach and tool combination in a real-world software development project, and providing a benchmark for comparison with the other evaluations to be performed. The Laboratory for Analysis and Architecture of Systems (LAAS) (LAAS 2006), of the French National Centre for Scientific Research (CNRS), was established in 1967 and is associated with three university institutions in Toulouse being Paul Sabatier University (UPS), the National Institute of Applied Science (INSA) and the National Polytechnic Institute (INP). As Part of the Engineering Sciences Department of CNRS, LAAS conducts research in four keys areas, namely 1) Micro and Nano Systems (MINAS), 2) Modelling, Optimization, and Control of Systems (MOCOSY), 3) Autonomous Robots and Systems (ROSA), and 4) Critical Information Processing Systems (SINC). There are over 500 researchers, engineers, and technical support personnel at LAAS, responsible for over 1400 research publications each year.

The project for this case study involved the design and development of a new online information system to support the process of submitting and reviewing technical reports by LAAS staff members and industry partners for publication in an internal quarterly journal. This project was selected for the case study because it was a real project available at the time of the research, and typical of the type and size of projects regularly conducted within the organisation, as recommended in (Kitchenham, B., Pickard & Pfleeger 1995). The case study participants included five people from the Systems Engineering and Integration (ISI) research group of LAAS, consisting of a second year PhD candidate designated by LAAS to be requirements analyst, a senior Professor as the customer representative, and three professional researchers as key end-users. The project team members were selected using the normal staffing procedure at LAAS, and had varying levels of familiarity and experience in working on software development projects and participating in collaborative requirements workshops, with the analyst having the least amount of expertise, and the customer representative having the most.

6.3.1 Procedures

The case study began with a thirty-minute training session on the MUSTER tool given in the morning to the requirements analyst by the researcher, followed immediately by a session to determine the scope for the case study project (i.e. a 'scoping session'), which lasted approximately one hour. This session was conducted in a small meeting room onsite at LAAS, with only the requirements analyst, customer representative, and researcher present. The requirements analyst and customer representative sat next to each other, both facing the screen of the laptop running the MUSTER tool, while the research sat behind the two project team members as a silent observer.

A session to determine the high-level requirements for the case study project (i.e. a 'high-level session') was performed in the afternoon of the same day, at the same location, in a larger meeting room, with the requirements analyst and customer representative being joined by the three key end-users. The project team members were seated around a 'U' shaped table arrangement to ensure they were all able to see the projection screen. The requirements analyst sat on one side of the arrangement in front of the laptop running the MUSTER tool, with the image projected on the screen for all to see. The researcher once again sat in the back of the room as a silent observer throughout the session. An introduction to this session was given by the requirements analyst, which included an overview of the system under investigation, and a brief demonstration of the MUSTER tool. The information elicited during the previous scoping session was reviewed out aloud by the requirements analyst with all the other project team members looking on. This introduction took approximately half an hour to complete, with the remaining two hours of the session spent entirely on eliciting requirements information for the target system using the MUSTER tool.

During the high-level session the researcher used a sheet (see Appendix G) to guide and record important and relevant observations. These comments were then grouped by identifiable themes and used to produce notes that would accurately represent the key actions of the participants and interactions with the tool. Directly after the

completion of this same second session, the five participants were given a questionnaire (see Appendix F) to complete and return anonymously. The questionnaire was used to collect feedback about the opinions and experiences of the participants on the tool they had just used. The results from all of the feedback questionnaires were then consolidated into a single form, and the combined team score for each statement and category of statements was calculated using frequency and a points system.

All of the introduction, training, and elicitation sessions were conducted in French, with French language versions of the approach and tool being used. The observation sheet and feedback questionnaire used for this evaluation were also both in French. All instruments used in this evaluation were originally developed in English, and then translated from English to French by the researcher. Similarly, all the results from this evaluation were originally produced in French, and then translated from French to English by the researcher. All of the translations that took place as part of this evaluation were thoroughly checked by a native French speaking researcher with excellent English language skills, who was not involved with the larger research project in any way. The ethical considerations detailed in Section 1.5 of this thesis were also applied to the planning, preparation, and performance of this evaluation.

6.3.2 Results

The following section details the results and subsequent analysis performed on the three data sources from the case study, being 1) the information elicited during the scoping and high-level sessions, 2) the observation sheet used by the researcher, and 3) the feedback questionnaire completed by the participants.

6.3.2.1 Elicited Information

During the scoping and high-level sessions, the requirements analyst was able to elicit from the stakeholders the information presented in Table 6.3.2.1 below, where each instance of any information type represents one piece of information. We can see that with one piece of information elicited about every two minutes during the scoping

session (60 minutes / 27 pieces of information = 1 piece of information every 2.2 minutes), and every one and half minutes during the high-level session (120 minutes / 72 piece of information = 1 piece of information every 1.6 minutes), both the tool and approach used would appear to have produced positive results. Although this data alone has only limited analysis value, it will be useful later for the purposes of comparison with the experiment performed at IUT (see Section 6.5).

Table 6.3.2.1: Summary of the results from the elicited information

Information Type	*Number of Instances*
Scoping Session	
System Goals	6
Project Constraints	4
Stakeholder Groups	5
Use Cases	12
High-level Session	
System Features	10
Functional Requirements	52
Non-functional Requirements	10

6.3.2.2 Observation Notes

During the high-level session it seemed virtually impossible for the participants to just elicit requirements without some degree of analysis. Throughout the session there was constant conversation, and although many points were in fact discussed in some detail, often nothing was actually recorded until a consensus had been reached. An example of this was a lengthy discussion about if the target system should accept PDF files and/or DOC files. It appears that those participants who had more technical experience and knew more about requirements engineering and the problem domain, were the ones who wanted to discuss and analyse points more, rather than just elicit and record them. In fact it was noted that the participant who talked the most during the session was in actuality the one who contributed the least amount of information that was physically recorded. This would tend to imply that lots of detailed technical discussion does not necessarily translate to lots of requirements information.

The initial rate of elicitation at the start of the session was relatively slow, however there were a number of short periods when the contribution of the participants, and the amount of information elicited, increased significantly. This appeared to happen around the thirty minute mark, and then again around the one hour mark. However when the group would begin to elicit the requirements for a new feature, once more the rate would be initially slow. For the first half an hour only three of the participants were contributing significantly to the output of the group, with one almost completely silent. This least active participant only seemed confident enough to join in after gaining a much better understanding of the approach that was being followed. It was also observed that the requirements analyst was so busy listening to the discussions and navigating the tool, that his actual contribution to the recorded information was very minimal.

In the general the participants rarely referred back to what they had already elicited as they continued to move forward in the process. The exception to this was that they often switched between information types, but this was only as a result of advice offered by the tool showing a possible link. The reason for this might be that the information already elicited was still fresh in their minds, and there was little or no need to review it again visually. Likewise, the participants did not often refer back to the advice they had previously been offered after it had been initially used. On the whole, the participants used the advice mainly at the beginning of each new elicitation task, and did not refer back to it regularly throughout the relevant task. Although this was not completely unexpected, it was anticipated that the more novice the analysts were, the more they would regularly refer back to the offered advice.

6.3.2.3 Feedback Questionnaire

The five questionnaire responses were consolidated into a single form, and the combined score was calculated for each statement and categories of statements, based on frequency analysis and a points system whereby each ‘Strongly agree’ was worth +2 points, each ‘Agree’ +1, each ‘Neutral’ 0, each ‘Disagree’ -1, and each ‘Strongly disagree’ was worth -2, in accordance with (Research Methods Knowledge Base

2006). As there were five responses in total, each statement then had a maximum (highest) possible score of +10 (if all the participants strongly agreed) and a minimum (lowest) possible score of –10 (if all participants strongly disagreed).

As can be seen in Table 6.3.2.2 below, the highest marked statements with a score of +7 out of a possible maximum of +10, meaning these were the statements that the participants most strongly agreed with, was that the tool was easy to understand (S01), the tool had a logical structure (S06), and the tool was easy to navigate (S10). The participants also agreed to the same degree that they would use the tool again (S22), they would recommend the tool to others (S24), and overall they were satisfied with the tool (S25). It is no coincidence then that of the six main categories of statements, understandability and learnability of the tool ranked the highest with a score of +18 out of a possible maximum of +30, followed closely by useability on +17. These results would suggest that the participants felt strongly that the tool was simple to understand, learn, and use, and that their overall satisfaction and acceptance of the tool was high.

We can also see that the lowest marked statements with a score of 0 out of possible maximum of +10, meaning these were the statements that the participants least strongly agreed with, and in this particular case were completely neutral to, was that the results of the tool were useful (S17), and that the results of the tool were of a good quality (S18). It is therefore no surprise that the effectiveness of the tool was the lowest ranking main category of statements with a score of +3 out of a possible maximum of +30. However the large number of neutral responses with respect to the quality and the usefulness of the results from the tool may be attributed to the fact that the participants were not prepared to make a judgement on this at such an early stage of the larger project without having seen their impact on the subsequent phases of development. That they were neutral as to the tool being better than others they had used may have also been as a result of their lack of experience. Also scoring low with marks of only +2 were the statements saying that the tool was better than others they had used (S23), and that the tool would improve the quality of results (S28).

Table 6.3.2.2: Summary of the results from the feedback questionnaire

Statement	**Strongly agree (+2)**	**Agree (+1)**	**Neutral (0)**	**Disagree (-1)**	**Strongly disagree (-2)**	**Total Score**
S01. The tool was easy to understand	3	1	1			+7
S02. The information was presented and organized in an way and format that is easy to understand	1	3	1			+5
S03. The tool was not overly complex	1	4				+6
Understandability:						**+18**
S04. The tool was easy to learn how to use		5				+5
S05. It was easy to remember how to do things in the tool	1	4				+6
S06. The tool had a logical structure and made sense to me	3	1	1			+7
Learnability:						**+18**
S07. The user interface of the tool was attractive	1	1	3			+3
S08. The layout and language of the tool was consistent	1	3	1			+5
S09. The layout and language of the tool was convenient	1	3	1			+5
Attractiveness:						**+13**
S10. The user interface of the tool was easy to navigate and move around	3	1	1			+7
S11. It was easy to find what I was looking for in the tool		5				+5
S12. The tool was easy to use	1	3	1			+5
Useability:						**+17**

S13. The tool was efficient to use	1	2	2			+4
S14. The tool helped me be productive		4	1			+4
S15. The response time of the tool was good	2	2	1			+6
Efficiency:						**+14**
S16. The tool was effective to use		3	2			+3
S17. The results of the tool were useful			5			0
S18. The results of the tool were of a good quality			5			0
EFFECTIVENESS:						**+3**
S19. The tool was flexible enough to be used in other system development projects	2	2	1			+6
S20. The tool has the necessary functions and features to support the task of requirements elicitation		3	2			+3
S21. I felt comfortable using the tool		4	1			+4
S22. I would use the tool again	3	1	1			+7
S23. The tool was better than others I have used for requirements elicitation	1		4			+2
S24. I would recommend the tool to others	2	3				+7
S25. Overall I am satisfied with the tool	2	3				+7
S26. Using the tool save me time	2	2	1			+6
S27. Using the tool save me effort	2	2	1			+6
S28. Using the tool would improved the quality of the results	1		4			+2
S29. My knowledge of requirements elicitation has increased by using the tool		5				+5

When asked at the end of the questionnaire to list the most positive aspects of the tool, three of the five participants mentioned the fact that the tool was both easy to use and understand, while three also suggested that the tool was very useful for the collection of information during group discussions and interactive workshops. Two of the

participants made note of the fact that the advice offered by the tool provided good help and support, and another two pointed out that the tool supplied a structured way of performing requirements elicitation. One participant cited the specific ability to enter information without lots of imposed constraints as a positive aspect of the tool and its operation.

When asked at the end of the questionnaire to list the most negative aspects of the tool, comments received from the participants included that the interface should be made more attractive with respect to the colours and forms used, and that there was no way to review a detailed history of all the actions and changes performed in the tool. Two of the five participants suggested that more questions needed to be added in the offered advice to provide more detailed help and guidelines for the users, and one specifically mentioned the necessity for more assistance in eliciting the relationships between Use Cases and Features. In order to be able to visualize requirements during the process of elicitation, one participant suggested that the tool should include a simple graphics feature that would enable users to draw Use Case and Data Flow diagrams during the workshops.

6.3.3 Discussion

Although this case study was conducted on a relatively small scale, we are able to identify a number of significant and important results. The case study produced significant rates of elicitation for both the scoping and high-level sessions across seven different information types, with 27 pieces of information elicited and recorded over one hour during the scoping session with only the analyst and customer present, and 72 pieces of information elicited over two hours during the high-level session with just five participants including the analyst. The difficulty in separating elicitation and analysis was clearly observed, particularly for the more technically minded participants. The rate of elicitation was seen to go through cycles during the session, but was often spurred on and encouraged by the advice offered by the tool. High marks were recorded in the feedback for Understandability, Learnability and Useability, with good scores also for the overall satisfaction and acceptance of the tool. However, it is interesting to note that the feedback showed the participants were

generally unsure as to the effect the tool had on the quality of the results, possibly because of their own interpretation of the term, or the lack thereof.

Despite the participants often engaging in detailed discussion and debate, a substantial amount of information was still elicited and recorded during the session, which would tend to support the received feedback that the tool was easy to use and efficient. However this is somewhat counter to the low result received for the overall effectiveness of the tool, although 12 out of 15 responses in this section of the questionnaire were neutral. Even with this relatively low mark for effectiveness, 4 of the 5 participants provided additional comments stating that the support offered by the tool was helpful, as was the structure it provided. The observation that most of the participants had no problem understanding the process of using the tool for requirements elicitation was also strongly supported by the feedback received. Furthermore it was observed that by the end of the session all of the participants had become very comfortable and confident with the requirements elicitation approach provided by the tool, which again compliments the feedback result where by all five of the participants agreed that their knowledge of requirements elicitation had increased by using the tool.

Validity and Reliability

In order to strengthen these results, a number of steps were taken to minimise the threats to validity and reliability through the careful design of the case study, including the instruments used and the processes for data collection and analysis (Stake 1995). Both the internal and external validity of the case study was enhanced by selecting a real project that was typical for the organisation, and by using the normal staffing procedure of the organisation to determine the case study participants, who were also real stakeholders of the system (Kitchenham, B. A. & Pickard 1998b). The range of participant experience and knowledge related to the case study activities, and their unfamiliarity with our tool and other tools that might be similar, were in fact desirable characteristics of the case study, rather than confounding factors. However as with all case studies it is difficult to determine to what extent the results can be reliably transferred and applied to other projects and domains, especially since our case study was conducted within a single organisation.

6.4 INSA Case Study Experiment

The next step in the evaluation of the OUTSET approach and MUSTER tool was a case study experiment within the scope of a Masters degree subject at the National Institute of Applied Sciences (INSA) in Toulouse, with the goal of evaluating the performance of our approach and tool combination in a software development project when performed by novice analysts in the absence of a prescribed methodology. Created in 1963, INSA Toulouse (INSA Toulouse 2006) is a higher education and research school dedicated only to engineering, and covering ten specializations including Biochemical, Civil, Electrical, Computer, Network, Mechanical, Industrial, and Systems Engineering, in addition to Mathematics and Physics. The facility spans 45 acres with 35 buildings, housing 2050 Masters of Engineering students, 215 permanent academic staff, and 238 administrative and technical staff. It is part of a French network of five INSA centres with locations also in Lyon, Rennes, Rouen, and Strasbourg, which represents the largest group for the training of engineers in France.

The case study experiment was integrated within the curriculum of an existing six-month project course, at the point in the project designated as 'Requirements Elicitation and Analysis'. During this course students are expected to design and document the electrical and electronic components of a system which is expected to run all aspects of an environmentally friendly (ecological) family-sized house. This project was selected as the case study experiment because it represented a real-world situation whereby novices were required to elicit requirements for a relatively complex system from other novices as both customer and user. The case study participants consisted of fourteen 2nd year Masters of Industrial Systems Engineering students from INSA, and were selected because all fourteen students were from the same class (one class of fourteen students in total), and all had approximately the same minor experience and expertise with respect to requirements elicitation and software engineering in general, (i.e. there were no mature-age students in the class who may have had industry experience).

6.4.1 Procedures

The case study experiment began by randomly dividing the fourteen participants into seven groups of two people each, as was required by the larger project of the degree subject, where each member was required to play the role of both analyst and customer/user. Each group was then assigned an individual PC workstation spread out around a large laboratory-style class room at INSA, in such a way, that no group could easily distract, hear, or interact with another. The groups were then given a unique login account and project ID number for a common MUSTER installation and configuration that had been setup on a network server. The case study experiment was conducted over a three-day period consisting of seven sessions of about one hour and fifteen minutes each, i.e. eight hours and forty-five minutes in total, all conducted in the same room. The first session involved a brief introduction where a number of important background concepts for the case study experiment were explained and discussed, as well as a demonstration and training on the OUTSET approach and MUSTER tool as a warm-up to the forthcoming elicitation sessions (Zowghi & Paryani 2003). The remaining six sessions were dedicated exclusively to eliciting requirements and requirements related information for the target system using the MUSTER tool and the embodied OUTSET approach.

During the six elicitation sessions the researcher used a sheet (see Appendix G) to guide and record observation as to the actions of the participants and interactions with the tool. This involved the researcher regularly circulating around the room, spending approximately equal amounts of time observing each group. The observations on all the groups across the sessions were combined into a single document and then grouped according to theme. After the final session, all the participants were asked to complete and return feedback questionnaire (see Appendix F) in order to collect details about the opinions and experiences of the participants on the tool they have just used, and the approach for requirements elicitation they had just followed. All of the responses from the feedback questionnaire were then consolidated into a single form, and the combined class score was calculated for each statement and category of statements using frequency analysis and a points system.

All of the introduction, training, and elicitation sessions were conducted in French, with French language versions of the approach and tool being used. The observation sheet and feedback questionnaire used for this evaluation were also both in French. All instruments used in this evaluation were originally developed in English, and then translated from English to French by the researcher. Similarly, all the results from this evaluation were originally produced in French, and then translated from French to English by the researcher. All of the translations that took place as part of this evaluation were thoroughly checked by a native French speaking researcher with excellent English language skills, who was not involved with the larger research project in any way. The ethical considerations detailed in Section 1.5 of this thesis were also applied to the planning, preparation, and performance of this evaluation.

6.4.2 Results

The following section details the results and subsequent analysis performed on the three data sources from the case study experiment, being 1) the information elicited during the six requirements elicitation sessions, 2) the observation sheet used by the researcher, and 3) the feedback questionnaire completed by the participants.

6.4.2.1 Elicited Information

During the six sessions the groups were able to elicit the information presented in Table 6.4.2.1 below, where each instance of any information type represents one piece of information. We can see from these results that each of the groups was able to successfully elicit instances for each of the information types, at an average rate of one piece of information elicited about every three and a half minutes (450 minutes / 125 pieces of information = 1 piece of information every 3.6 minutes). This result was achieved with each group having only two people, and one of those was also responsible for entering the information into the tool.

Table 6.4.2.1: Summary of the results from the elicited information

Group	Goals	*Constraints*	*Use Cases*	*Features*	*FR*	*NFR*
1	7	14	17	19	48	12
2	9	7	13	22	57	16
3	9	10	12	19	41	11
4	10	8	34	19	55	14
5	15	11	24	37	48	14
6	19	10	25	20	49	9
7	11	5	21	18	52	6
Total:	80	65	146	154	350	82
Average:	11.43	9.29	20.86	22.00	50.00	11.71

6.4.2.2 Observation Notes

During the case study experiment it was observed that the understanding of the participants, with respect to the approach for requirements elicitation and the various tasks that it entails, improved dramatically over the first two sessions to a degree where most of the groups understood what they were expected to do with respect to the overall activity. By the time the participants came to the stage of eliciting features for the system, they appeared to demonstrate a good comprehension of the approach and its relationship with the operation of the tool. The points in the process that appeared to cause the most confusion among the participants were in understanding the differences between functional and potentially non-functional requirements, and the possible links between use cases and features.

The participants in general appeared to find the tool very easy to use right from the first elicitation session immediately after their initial training. No additional questions were asked by any of the participants during the sessions with respect to the operation of the tool. The entry of elicited information into the tool, and the navigation between the various elicitation tasks, seemed to be particularly simple to learn and remember for the participants. One area of difficulty in using the tool that appeared to cause problems for some of the participants was the limitation of being able to view the entries of only one information type at a time. This required the participants to

regularly switch back and forth between the current and completed tasks in order to refer and reference back to information already elicited and entered.

Overall it seemed that by simply providing a structured approach by which to conduct the activity of requirements elicitation, the tool proved very helpful to the participants throughout the sessions. Of particular usefulness appeared to be the examples and suggestions given by the tool for each of the information types that required elicitation, as well as the way in which the tasks were presented and organized in a collapsible tree structure. However, as mentioned in the observation notes, there was still some confusion among the participants as to differences and relationships between information types. A significant number of tool's features were not used at all by the participants during the elicitation sessions. This was to be expected given that not all the functionality was addressed during the training, nor was there a need to utilize much of the possible operations of the tool given the scale of the case study experiments and the scope of the project.

Advice offered by the tool was mainly used by the participants at the beginning of each new task, as a way to understand the relevant background concepts and kick-start that particular part of the elicitation approach. It also seemed that the advice offered by the tool was always at least reviewed by the participants, and that for the most part the participants accepted and adopted it under the assumption that it was totally correct. As a result the assistance provided by the tool very much determined the actions of the participants and the course of eliciting information during the sessions. It is however important to note that only occasionally did the participants refer back to the advice previously offered by the tool for a particular task during each session. This could imply that the advice offered by the tool was only useful as a general introduction to each task, or that the advice offered by the tool at the start of each task was sufficient enough for the participants to complete it without further reference.

Each of the participants appeared to contribute approximately equally to the resultant output of their respective groups, however this may have been as a result of each team only having two members. There seemed to be good interaction between the participants in all of the groups, with several periods of rich brainstorming and fruitful discussion occurring in each session. Detailed analysis of any individual issue

appeared to occur only when there was a direct conflict in understanding or opinion between the members of the group. Although the participants often examined these issues for extended periods of time, it was interesting to observe that despite the considerable length of time they spent eliciting information, only rarely did conversations deviate onto topics unrelated to the task at hand. This may have been as a result of the participants being particularly dedicated to the task at hand, and/or the tool being particularly effective at keeping the participants focused on and committed to the process of eliciting requirements.

6.4.2.3 Feedback Questionnaire

The fourteen questionnaire responses (one from each of the two participants from all seven of the groups) were consolidated into a single form, and the combined scores was calculated for each statement and categories of statements, based on frequency analysis and a points system whereby each ‘Strongly agree’ was worth +2 points, each ‘Agree’ +1, each ‘Neutral’ 0, each ‘Disagree’ -1, and each ‘Strongly disagree’ was worth -2, in accordance with (Research Methods Knowledge Base 2006). As there were fourteen responses in total, each statement then had a maximum (highest) possible score of +28 (if all the participants strongly agreed) and a minimum (lowest) possible score of –28 (if all the participants strongly disagreed).

As can be seen in Table 6.4.2.2 below, the highest marked statements with scores of +28, +26, and +24 respectively out of a possible maximum of +28, meaning that at least 10 of the 14 participants strongly agreed with the statements, was that the tool was easy to understand (S01), the tool was easy to learn (S04), and it was easy to remember how to do things in the tool (S05). Other statements that at least half of the participants strongly agreed with included that the tool was not overly complex (S03), and the tool was easy to use (S12). Consequently the highest ranking main categories of statements were once again those concerning the understandability of the tool with a score of +70 out of a possible maximum of +84, followed by those relating to the learnability of the tool with a score of +62. The results also show that 11 of the 14 participants agreed that the approach was effective, and 9 agreed that the approach was efficient. Ten agreed that using the approach had increased their knowledge of

requirements elicitation, and half of the participants agreed that they were overall satisfied with the tool, with another three strongly agreeing to the same statement.

We can also see that the lowest marked statement with a score of +2 out of a possible maximum of +28, meaning that this was the statement the participants least strongly agreed with, was that the tool was better than others they had used (S23). However this can be attributed to the 10 neutral responses to this statement, which may once again imply a lack of participant experience with requirements elicitation tools. The next lowest scoring statements related to the attractiveness of the user interface of the tool (S07) with a mark of +5 out of +28, and the tool being responsible for saving the participants time (S26) with +7. Interestingly, attractiveness was the main category of statements that ranked the lowest of the six with a score of +27 out of a possible maximum of +84, suggesting that of all the different aspects of the tool, including efficiency and effectiveness, it was the look and layout of the tool which was the least successful. It is however important to note that each of the statements scored a positive result overall, and only 15 negative responses were received out of a total of 406, equating to less than 4 percent.

Table 6.4.2.2: Summary of the results from the feedback questionnaire

Statement	**Strongly agree (+2)**	**Agree (+1)**	**Neutral (0)**	**Disagree (-1)**	**Strongly disagree (-2)**	**Total Score**
S01. The tool was easy to understand	14					+28
S02. The information was presented and organized in an way and format that is easy to understand	6	8				+20
S03. The tool was not overly complex	8	6				+22
Understandability:						**+70**
S04. The tool was easy to learn how to use	12	2				+26
S05. It was easy to remember how to do	10	4				+24

	things in the tool					
S06.	The tool had a logical structure and made sense to me	1	10	3		+12
	Learnability:					**+62**
S07.	The user interface of the tool was attractive	1	6	4	3	+5
S08.	The layout and language of the tool was consistent	2	7	5		+11
S09.	The layout and language of the tool was convenient	3	5	6		+11
	Attractiveness:					**+27**
S10.	The user interface of the tool was easy to navigate and move around	4	5	3	2	+11
S11.	It was easy to find what I was looking for in the tool	6	4	4		+16
S12.	The tool was easy to use	7	7			+21
	Useability:					**+48**
S13.	The tool was efficient to use	2	7	5		+11
S14.	The tool helped me be productive	2	5	7		+9
S15.	The response time of the tool was good	5	8	1		+18
	Efficiency:					**+38**
S16.	The tool was effective to use	1	10	3		+12
S17.	The results of the tool were useful	2	7	5		+11
S18.	The results of the tool were of a good quality	2	6	5	1	+9
	Effectiveness:					**+32**
S19.	The tool was flexible enough to be used in other system development projects	3	6	5		+12
S20.	The tool has the necessary functions and features to support the task of	3	6	2	3	+9

	requirements elicitation					
S21.	I felt comfortable using the tool	7	6	1		+20
S22.	I would use the tool again	3	5	6		+11
S23.	The tool was better than others I have used	2		10	2	+2
S24.	I would recommend the tool to others	2	6	6		+10
S25.	Overall I am satisfied with the tool	3	7	4		+13
S26.	Using the tool save me time	2	5	5	2	+7
S27.	Using the tool save me effort	3	8	3		+14
S28.	Using the tool would improved the quality of the results	2	7	5		+11
S29.	My knowledge of requirements elicitation has increased by using the tool	2	8	2	2	+8

When asked to list the three most positive aspects of the tool, all fourteen of the participants stated that the tool was either simple to use, learn, or understand. Seven of the participants made reference to the advice offered by the tool as being helpful, useful or educational, and two noted that the tool provided improvements in terms of the number of requirements elicited and the efficiency of the process. The fact that the tool enables users to enter and modify all the information elicited at anytime was mentioned by three of the fourteen participants, and two more pointed out that because the tool was web-based it was therefore accessible from anywhere. Finally, four of the participants cited that providing a structured and logical approach for the elicitation of requirements was by itself a positive aspect of the tool, while three more comments were received suggesting that the approach prescribed by the tool promoted the elicitation of the maximum number of requirements.

When asked to list the three most negative aspects of the tool, ten of the fourteen participants suggested that the interface design and the colours in particular were either not nice or bad. Nine participants also made mention of the fact that the tool does not enable users to re-sort or re-organize the dynamic task list and the individual type-specific lists of elicited information. One participant noted that it is not possible

to have two or more windows of the tool open at the same time, and two implied that the tool would be better if there was a greater use of keyboard and mouse shortcuts. In addition, two of the fourteen participants recommended that the tool should allow functional requirements to be copied easily between features, and another proposed that the facility to create new information hierarchies, and generate custom links between instances of the different information types, was needed in the tool. Lastly, three participants felt that the tool might in fact be too simple (i.e. not sophisticated enough) for the elicitation of requirements for very complex systems.

6.4.3 Discussion

Once again we saw relatively high amounts of elicited information across all 6 information types, with an average of 125.29 pieces of information per group over the 6 one and a quarter hour sessions with only two members. From the observations we found that the tool appeared to be both easy to use, and useful in providing a way of working through the various tasks involved in the elicitation of requirements. Furthermore, the tool and specifically the support offered, seemed to focus the participants on the task at hand, and continued to drive both the participants and process forward. In the feedback, the tool received positive comments and high marks across the board for understandability, learnability, and useability once again, with all fourteen of the participants strongly agreeing that the tool was easy to understand. The area that received both the lowest marks and the most negative comments, with ten out of the fourteen participants making special mention of it, was the attractiveness of the user interface. However it is important to remember that attractiveness is subjective, and very much dependent on personal tastes, particularly in the absence of a formal definition, as was the case with the feedback questionnaire used.

Similarly high levels of correlation between the elicited information and the observation and feedback data were seen in this evaluation as in the previously presented case study, including the consistently large amounts of information elicited across the groups, and the general consensus that the tool was easy to learn and use. However we also saw the same mismatch between the amounts of information elicited and the positive effect the tool was observed to have on driving the process, with the

relatively low scores received for effectiveness in the feedback. Once again this was due to a large number of neutral responses to the statements in that section of the questionnaire, which could imply the participants do not fully understand the intended meaning of those statements. Not only was it observed that that participants appeared very comfortable using the tool, but this was backed up by the high marks received in the feedback for a statement along the same lines. Understanding of the requirements elicitation approach was observed to improve dramatically during the sessions, and this was supported by the fact that 10 of the 14 participants said in the feedback that their knowledge had increased as a result of using the tool.

Validity and Reliability

As was also the situation with case study presented in the previous section, a number of efforts were made to reduce the threats to the validity and reliability of the case study experiment. The project was selected based not only on the fact that it was already part of a larger established project, but because it was also typical of the size and scope of project that a novice analyst might be expected to participate in. Furthermore it was one where the participants would be sufficiently familiar with the kind of system under investigation, and the operations that the system would need to address. Participants were not only potentially real stakeholders of the target system, by being residents of ecologically sound homes in the future, but also appropriately qualified by their experience and expertise for the case study experiment as novice requirements analysts and existing users of similar systems in their own homes. Despite the results from this case study experiment being somewhat limited in their application, as no basis for comparison was available, we are able to later draw some useful conclusions on the useability of the tool for a domain different to that of the case study and experiment presented in the following section.

6.5 IUT Experiment

The last step in the evaluation of the OUTSET approach and MUSTER tool was an experiment within the scope of a Bachelors degree subject at the University Institute of Technology (IUT), Blagnac, with the goal of evaluating the relative performance of both our approach and tool in a controlled setting when used by novice analysts. IUT Blagnac (IUT Blagnac 2006) was established in 1974 as part of the University of Toulouse II - Mirail (UTM), and consists of three departments being Information Technology, Industrial Engineering and Maintenance, and Networks and Telecommunications, and offers three undergraduate technical diplomas (DUTs) in these same areas. The institute also provides four postgraduate professional licences (LPs) in Analysis and Programming, Aeronautical Maintenance, Mobile Networks and Security, and Installation and Maintenance Control Inspection.

The project used for this experiment involved the design and development of a new online information system to support the process of IUT students electronically submitting assignments and exams, and lecturers collecting, marking, and returning them. This project was selected for the experiment because it was a real potential system under initial investigation by IUT at the time, and was sufficiently similar in subject and scope to that of the case study project conducted at LAAS to allow for some high-level comparisons, in addition to it being one that would be appropriately familiar and useful to the participants. The experiment participants consisted of thirty 3rd year Bachelor of Computer Science students from IUT. The participants were selected because all thirty students were from the same course (two classes of fifteen students in each), and all had approximately the same limited experience and expertise with respect to requirements elicitation and software engineering in general (i.e. there were no mature-age students who may have had industry experience in either of the classes).

6.5.1 Procedures

The experiment began by randomly dividing the fifteen participants from the first class into three groups of five members each using the alphabetically sorted class list. Each group was then randomly assigned to one of three available tools spread out around a large laboratory at IUT, being 1) a fully functional version of the MUSTER tool with all the available advice (hereafter referred to as 'FULL'), 2) a fully functional version of the MUSTER tool with only part of the advice provided (hereafter referred to as 'PART'), and 3) a manual (non-automated) version of the MUSTER tool where all the available advice and data formats were provided in a structured Word document (hereafter referred to as 'MANU'). One member from each of the groups was randomly assigned the responsibility of playing the analyst, and the remaining group members were assigned the role of customer and user stakeholders. The researcher then gave a brief introduction to all the participants whereby the project and the system to be investigated was introduced, and the procedure for the session was explained. Following this each group was trained individually on their respective tool for fifteen minutes by either the researcher, or one of the two additional observers who were fellow researchers from LAAS that were extensively briefed on the research project and their role in the experiment.

The primary goals (3), key constraints (3), and core use cases (5) for the target system were provided to the groups, and reviewed out aloud by the researcher with the entire class. This information was given to the groups in order to provide a more complete background picture of the target system for their elicitation activities, and also to enable a better comparison with the high-level session of the LAAS case study evaluation where the goals, constraints, and use cases were also pre-specified. The groups were then given one hour uninterrupted to elicit features, functional requirements, and non-functional requirements for the target system. During the session the researcher took detailed notes via an observation sheet (see Appendix G), which were then documented, grouped by theme, and cross-validated the following day individually by the two additional observers involved in the experiment. Upon completion of the session, all the participants were asked to complete a feedback questionnaire (see Appendix F) in order to collect details on the opinions and

experiences of the participants about the tool they have just used, and the approach for requirements elicitation that had just followed. The entire process explained above was repeated again immediately after with a second class containing the same number of students, the same tools, and the same procedures.

At the same time a week later the groups in the first class were asked to review a complete but randomised list of all the information elicited by their entire class during the session the previous week, and provide feedback as to which features, functional requirements, and non-functional requirements they thought were valid (i.e. they believed that the elicited piece of information was relevant to the case study project), invalid (i.e. they believed that the elicited piece of information was irrelevant to the case study project), or unsure about (i.e. they were unsure if the elicited piece of information was relevant or irrelevant to the case study project). The purpose of this 'cross validation process' was therefore to decide which pieces of elicited information were actually relevant to the case study project, in order to determine the relative effectiveness of the three available tools, and as such, was again repeated exactly and immediately after with the second class.

All of the introduction, training, and elicitation sessions were conducted in French, with French language versions of the approach and tool being used. The observation sheet and feedback questionnaire used for this evaluation were also both in French. All instruments used in this evaluation were originally developed in English, and then translated from English to French by the researcher. Similarly, all the results from this evaluation were originally produced in French, and then translated from French to English by the researcher. All of the translations that took place as part of this evaluation were thoroughly checked by a native French speaking researcher with excellent English language skills, who was not involved with the larger research project in any way. The ethical considerations detailed in Section 1.5 of this thesis were also applied to the planning, preparation, and performance of this evaluation.

6.5.2 Results

The following section details the results and subsequent analysis performed on the three data sources from the experiment, being 1) the information elicited during the requirements elicitation sessions, 2) the observation sheets used by the observer, and 3) the feedback questionnaire completed by the participants.

6.5.2.1 Elicited Information

Table 6.5.2.1 below shows the results before and after the cross validation process of the first class, the second class, and the two classes combined. In the first part of the table, we can see the number of features (‘F’), functional requirements (‘FR’), and non-functional requirements (‘NFR’) elicited by the three groups of the first class (i.e. group ‘1’ with the ‘FULL’ tool, group ‘2’ with ‘PART’ tool, and group ‘3’ with the ‘MANU’ tool), both before (‘Before’) and after (‘After’) the cross validation process. In the second part of the table, we can see the same results for the three groups of the second class, and in the last part of the table, we can see the results of the three groups of the first class combined with the three groups of the second classes based on the tool each group was assigned and used.

For the process of cross validation, the rule was applied that if any group from the same class marked a piece of information as invalid, including the group who originated it, then that piece of information was removed from the list. If two groups from the class marked any piece of information as unsure, then that piece of information was also removed from the list. In the table below the numbers in square brackets represent the average number of functional requirements elicited per feature, and the figures in round brackets show the number of pieces removed by the originating group, and the number of pieces removed by the other groups in the class.

Table 6.5.2.1: Summary of the results from the elicited information

Group	*Before*			*After*		
	F	*FR*	*NFR*	*F*	*FR*	*NFR*
Class 1						
1 FULL	11	28 [2.55]	7	8 (1/3)	22 [2.75]	6 (0/1)
2 PART	9	23 [2.56]	4	7 (0/2)	20 [2.86]	2 (1/1)
3 MANU	11	35 [3.18]	8	7 (0/4)	23 [3.29]	3 (2/5)
Class 2						
4 FULL	10	30 [3.00]	4	10 (0/0)	30 [3.00]	3 (0/1)
5 PART	4	20 [5.00]	9	3 (1/1)	14 [4.67]	4 (0/5)
6 MANU	12	29 [2.42]	8	7 (2/5)	18 [2.57]	4 (2/4)
Combined (Class 1 + Class 2)						
1+4 FULL	21	58 [2.76]	11	18 (1/3)	52 [2.89]	9 (0/2)
2+5 PART	13	43 [3.31]	13	10 (1/3)	34 [3.40]	6 (1/6)
3+6 MANU	23	64 [2.78]	16	14 (2/9)	41 [2.93]	7 (4/9)

From the results of the first class before the cross validation, we can see that the manual tool (Group 3) had more functional requirements than the fully functional tool (Group 1) but the same number of features and only one more non-functional requirement. The partial advice tool (Group 2) on the other hand had the least number of pieces for all the information types. After the cross validation we can see that the fully functional tool now has the most number of features and non-functional requirements, and only one functional requirements less than the manual tool. The partial advice tool still had the least number of functional and non-functional

requirements, but the same number of features as the manual tool. We can therefore suggest from this that although the manual tool initially produced the most amount of information (i.e. more efficient), the fully functional tool produced more relevant information (i.e. more effective) as shown by the cross validation, with 4 features and 12 of the functional requirements elicited using the manual tool subsequently removed from the list.

From the results of the second class before validation, we can see that the manual tool (Group 6) had significantly more non-functional requirements, and only one less non-functional requirement than the fully functional tool (Group 4) but more features. Once again the partial advice tool (Group 5) had the least number of features and functional requirements. After the cross validation we can see that the fully functional tool again now has the most number of features, and the most number of functional requirements, with all three of the groups having roughly the same number of functional requirements. Once more the partial advice tool had the least number of functional requirements, as well as the least number of features. Therefore as was also the case in the first class, the manual tool initially produced the most amount of information during the elicitation session, and after cross validation the fully functional tool again had produced the most amount of relevant information, with 5 features and 11 functional requirements removed from the list of information elicited by the manual tool.

When we compare the results of first and the second class, it is not surprising that in both cases the total number of pieces of elicited information after cross validation rank the fully functional tool first, followed by the manual tool, and then the partial advice tool last. We can also see that the manual tool recorded the highest average number of functional requirements per feature both before and after the cross validation in the first class, and in the second class the partial advice tool had the highest average number of functional requirements per feature also before and after the cross validation. Therefore, given that the tool with the highest average functional requirements per feature in each class maintained this position even after the cross validation, but the tool with the highest average was in fact different for the two classes, this would tend to indicate that the number of functional requirements per feature is more a characteristic of the way in which the group worked (i.e. the

attention given to each feature during the elicitation of functional requirements), rather than a feature of the tool they were assigned and used.

Because exactly the same procedures and conditions were used for the elicitation sessions of both the classes, it is possible to analyse the results from the two classes combined. From this we can see that the manual tool not only produced the most amount of information in total before the cross validation (i.e. most efficient), but also produced the most amount of information for each of the three different information types. Likewise the fully functional tool delivered the most amount of information in total after the cross validation (i.e. more effective), in addition to the most features, functional requirements, and non-functional requirements individually as well. When we look at the pieces of information removed as a result of the cross validation process, we can see that the two groups using the manual tool were responsible for invalidating one third of the total features and functional requirements removed from their own list, i.e. 6 out of 18 features, whereas the two groups using the fully functional tool removed only one fifth of their own features, i.e. 1 out of 5. This would tend to imply that even those groups that used the manual tool agreed that it produced the greatest number of irrelevant pieces of information.

6.5.2.2 Observation Notes

During the session most of the groups started each new elicitation task quite slowly but generally got faster as they became more involved and confident. Several of the groups seemed to lose focus about half way through the experiment (i.e. around the 30 minute mark) with just two or three members continuing to work actively for the remaining time. A number of times during the session some members of the groups engaged in discussion on topics completely unrelated to the experiment, which would often temporarily distract the other members from the task at hand.

All of the groups found it difficult to just elicit information, but instead often discussed and analysed ideas in detail. It was also noticed that in order to elicit functional requirements for the features of the new system, several groups tried to visualize possible screens. Therefore, it seemed that the use of some analysis and

design techniques can actually help in the development of requirements during elicitation, demonstrating the close and potentially constructive relationship between both elicitation and analysis, and Requirements Engineering with design.

All three of the experiment observers identified that most of the groups found eliciting features for the system the easiest when compared to functional and non-functional requirements. Several groups appeared to have difficulty understanding the conceptual difference between functional and non-functional requirements, and subsequently found eliciting non-functional requirements the hardest.

It was noted during the sessions that if one or two of the group members were highly motivated, then this had a significant and positive impact on the amount of effort expended by the overall group, however it was difficult to determine whether the groups that were more motivated used the advice offered by the tool any more or less than the groups that were less motivated. In general, the effort of the groups seemed to also be better if they had a leader that was strong and committed to the work, i.e. leadership made a significant difference to the commitment of the other members. This once again demonstrates the strong influence of social factors on elicitation, and requirements workshops in particular.

6.5.2.3 Feedback Questionnaire

The five questionnaire responses from each of the groups for both of the classes were consolidated into a single form, and the combined scores was calculated for each statement and categories of statements, based on frequency analysis and a points system whereby each ‘Strongly agree’ was worth +2 points, each ‘Agree’ +1, each ‘Neutral’ 0, each ‘Disagree’ -1, and each ‘Strongly disagree’ was worth -2, in accordance with (Research Methods Knowledge Base 2006). As there were five responses for each group, each statement then had a maximum (highest) possible score of +10 (if all the members of the group strongly agreed) and a minimum (lowest) possible score of –10 (if all the members of the group strongly disagreed). The results for each of the statements for the groups combined based on the tool they

used would therefore have a maximum and minimum possible score of +20 and –20 respectively.

As can been seen in Table 6.5.2.2 below, the most obvious result from the feedback questionnaire was that in 23 of the 29 statements, the fully functional version of the tool received the highest or equal highest combined group score, with the maximum losing margin for the remaining six statements being only +2 points out of a maximum possible difference of 40 points. Furthermore the fully functional tool received the highest combined group score for all six of the main categories being understandability, learnability, attractiveness, useability, efficiency, and effectiveness. In fact the fully functional tool received more than three times the number of points for effectiveness when compared to the manual tool (+37 versus +12), and almost three times again for attractiveness (+28 versus +10). With respect to the general useability of the tools, once more the fully functional tool received more than double the points given to the manual tool (+40 versus +17).

The highest combined group score received for any one of the statements was for the fully functional tool with +18 out of a possible maximum of +20 points in relation to its responsiveness (S15), although there was little chance of the manual tool scoring highly for this statement given its non-automated nature. The statement with the highest score for all six groups combined related to simplicity of the tools from an understanding perspective (S03). The lowest combined group score received for any one of the statements was for the manual tool with –3 points for the attractiveness of the interface (S07) which again is not surprising, although this was also the lowest scoring statement for all three of the tools and therefore overall, with the partial advice tool receiving only +2 points from both groups combined, and the fully functional tool +4. It is also interesting to note that all three of the tools scored relatively high and almost exactly the same for the participants' overall satisfaction with using the tool (S25), and the degree to which their knowledge of requirements elicitation was increased by using the tool (S29).

Table 6.5.2.2: Summary of the results from the feedback questionnaire

Statement	**Group 1 (FULL)**	**Group 2 (PART)**	**Group 3 (MANU)**	**Group 4 (FULL)**	**Group 5 (PART)**	**Group 6 (MANU)**	**Groups 1 + 4 (FULL)**	**Groups 2 + 5 (PART)**	**Groups 3 + 6 (MANU)**
S01. The tool was easy to understand	**+7**	**+6**	**+5**	**+8**	**+7**	**+3**	**+15**	**+13**	**+8**
S02. The information was presented and organized in a way and format that is easy to understand	**+3**	**+5**	**+1**	**+7**	**+7**	**+4**	**+10**	**+12**	**+5**
S03. The tool was not overly complex	**+7**	**+7**	**+7**	**+9**	**+8**	**+4**	**+16**	**+15**	**+11**
Understandability:	**+17**	**+18**	**+13**	**+24**	**+22**	**+11**	**+41**	**+40**	**+24**
S04. The tool was easy to learn how to use	**+6**	**+8**	**+6**	**+10**	**+8**	**+3**	**+16**	**+16**	**+9**
S05. It was easy to remember how to do things in the tool	**+5**	**+7**	**+8**	**+9**	**+5**	**+5**	**+14**	**+12**	**+13**
S06. The tool had a logical structure and made sense to me	**+5**	**+5**	**+6**	**+8**	**+7**	**+6**	**+13**	**+12**	**+12**
Learnability:	**+16**	**+20**	**+20**	**+27**	**+20**	**+14**	**+43**	**+40**	**+34**
S07. The user	**-2**	**0**	**-3**	**+6**	**+2**	**0**	**+4**	**+2**	**-3**

	interface of the tool was attractive								
S08.	The layout and language of the tool was consistent	**+5**	**+5**	**+2**	**+8**	**+4**	**+4**	**+13**	**+9**	**+6**
S09.	The layout and language of the tool was convenient	**+3**	**+4**	**+4**	**+8**	**+4**	**+3**	**+11**	**+8**	**+7**
	Attractiveness:	**+6**	**+9**	**+3**	**+22**	**+10**	**+7**	**+28**	**+19**	**+10**
S10.	The user interface of the tool was easy to navigate and move around	**+5**	**+9**	**0**	**+10**	**+3**	**+3**	**+15**	**+12**	**+3**
S11.	It was easy to find what I was looking for in the tool	**+2**	**+8**	**+1**	**+9**	**+5**	**+3**	**+11**	**+13**	**+4**
S12.	The tool was easy to use	**+7**	**+7**	**+5**	**+9**	**+7**	**+5**	**+16**	**+14**	**+10**
	Useability:	**+14**	**+24**	**+6**	**+26**	**+15**	**+11**	**+40**	**+39**	**+17**
S13.	The tool was efficient to use	**+4**	**+3**	**+4**	**+9**	**+6**	**+5**	**+13**	**+9**	**+9**
S14.	The tool helped me be productive	**+3**	**+2**	**+4**	**+5**	**+4**	**+5**	**+8**	**+6**	**+9**
S15.	The response time of the tool was good	**+8**	**+7**	**+2**	**+10**	**+5**	**+6**	**+18**	**+12**	**+8**
	Efficiency:	**+15**	**+12**	**+10**	**+24**	**+15**	**+16**	**+39**	**+27**	**+26**
S16.	The tool was	**+4**	**+3**	**+2**	**+9**	**+4**	**+3**	**+13**	**+7**	**+5**

	effective to use								
S17.	The results of the tool were useful	**+6**	**+3**	**+1**	**+6**	**+4**	**+2**	**+12**	**+7**	**+3**
S18.	The results of the tool were of a good quality	**+6**	**+4**	**+1**	**+6**	**+3**	**+3**	**+12**	**+7**	**+4**
	Effectiveness:	**+16**	**+10**	**+4**	**+21**	**+11**	**+8**	**+37**	**+21**	**+12**
S19.	The tool was flexible enough to be used in other system development projects	**+6**	**+4**	**+3**	**+6**	**+5**	**+4**	**+12**	**+9**	**+7**
S20.	The tool has the necessary functions and features to support the task of requirements elicitation	**+5**	**+6**	**+2**	**+10**	**+5**	**+5**	**+15**	**+11**	**+7**
S21.	I felt comfortable using the tool	**+4**	**+5**	**+6**	**+6**	**+5**	**+1**	**+10**	**+10**	**+7**
S22.	I would use the tool again	**+5**	**+7**	**+5**	**+6**	**+5**	**+4**	**+11**	**+12**	**+9**
S23.	The tool was better than others I have used	**0**	**+2**	**+2**	**+6**	**+3**	**+1**	**+6**	**+5**	**+3**
S24.	I would recommend the tool to others	**+5**	**+4**	**+2**	**+5**	**+5**	**+2**	**+10**	**+9**	**+4**
S25.	Overall I am satisfied with the tool	**+6**	**+5**	**+5**	**+5**	**+5**	**+5**	**+11**	**+10**	**+10**

S26.	Using the tool save me time	**+3**	**+5**	**+4**	**+6**	**+5**	**+4**	**+9**	**+10**	**+8**
S27.	Using the tool save me effort	**+5**	**+6**	**+3**	**+7**	**+8**	**+2**	**+12**	**+14**	**+5**
S28.	Using the tool would improved the quality of the results	**+2**	**+5**	**+3**	**+6**	**+4**	**+1**	**+8**	**+9**	**+4**
S29.	My knowledge of requirements elicitation has increased by using the tool	**+5**	**+5**	**+5**	**+4**	**+4**	**+3**	**+9**	**+9**	**+8**

When Group 1 and Group 4 were asked to list the three most positive aspects of the fully functional version of the tool they had just used, eight out of the combined ten members noted that the tool was easy to use, and nine mentioned the way in which the tool provided a structure for the arrangement of information. In addition, half of the members cited a gain in productivity as being a major positive aspect of the tool. When the same two groups were asked to list the three most negative aspects, seven out of the combined ten members stated that the interface was not attractive, and in particular the colours and icons used. Half of the members suggested that even more advice from the tool would be beneficial, however two members also recommended that it should be possible to hide the offered advice.

When Groups 2 and 5 were asked to list the three most positive aspects of the partial advice version of the tool they had just used, all ten of the combined members made reference to the fact that the tool was either simple or easy to use. Five of the participants stated that the advice offered by the tool for specific tasks was useful, and three commented that the tool had a logical organisation and arrangement. When the same two groups were asked to list the three most negative aspects, three of the combined ten members made reference to the presentation or interface of the tool. Three members proposed that more specific help was required with respect to the

steps in each task, and one member implied that a more detailed explanation on the overall approach for requirements elicitation was needed at the beginning.

When Group 3 and Group 6 were asked to list the three most positive aspects of the manual version of the tool they had just used, seven of the ten combined members declared that the tool was easy to understand, five commented on the presentation and arrangement of information, and three members cited the support the tools provided for the process of requirements elicitation. When the same two groups were asked to list the three most negative aspects, eight of the ten combined group members stated that the user interface of the tool was not attractive. Five of the members said that it was either not easy or difficult to navigate the tool through the process, and four suggested that more explanations and help in using the tool were needed.

6.5.3 Discussion

The most important and significant result from the experiment presented in this section was that in both of the classes the manual tool was more efficient by producing the most amount of information, but it was the fully functional tool that had the most after cross validation, and was therefore deemed to be most effective by producing the most amount of relevant information. The number of functional requirements per feature appeared to be unrelated to the tool used for the process of elicitation, but more a characteristic of the group in so far as how much time and effort they would allocate to each feature before moving on to the next. It was observed that the output of the groups seem to go through cycles during the session, and was significantly and positively impacted on by the commitment and motivation of just two or three members, and by new advice offered by the tool. From the feedback received we can see that all three of the tools were deemed to be not overly complex, as this was the statement with the highest combined marks. Likewise all three interfaces of the tools were reasoned to be not attractive, as this was the lowest scoring statement for each. In particular the manual tool received the lowest marks and the most negative comments regarding the attractiveness of the tool, as was to be expected. In all the six of the main useability categories, the fully functional tool

received the most positive feedback, followed most often by the partial advice, and then the manual tool.

Although there was some inconsistency in the commitment of the members throughout the session, each group still managed to elicit and record a relatively high amount of information for each of the different information types. Despite the manual tool receiving the lowest marks for efficiency, we have seen that it actually produced the most amount of information initially at least. However after the cross validation we do see some correlation in the data for the fully functional tool where it produced the highest amount of information and appropriately received the highest score for effectiveness. Although the fully functional tool did receive numerous positive comments in the feedback related to improved productivity, and got the most when compared to the partial advice and manual tool, even it scored relatively low for efficiency and effectiveness overall, indicating that there may have been some degree of misunderstanding with these statements in the questionnaire. Regardless of the tool they were assigned, all groups appeared to be overall satisfied with their experience, both from the noted observations and the feedback received. Likewise all the groups seemed to gain knowledge about requirements elicitation by using their tool, which was an observation also supported by the results from the questionnaire.

The careful design of this experiment allowed us to evaluate at the same time the efficiency and effectiveness of both the tool, by comparing fully functional and manual versions of the tool with the same process, and the approach, by comparing the fully functional and partial advice versions of the same tool. We are also able to make some interesting crossover conclusions as to the real-world benefits of the tool and approach through comparison with the previously presented case study. This is because of the close similarity between the two projects they were based on, and the similarities between the conditions under which they were conducted.

Validity and Reliability

The fact that students were used for this experiment, and all had similar experience and expertise, was actually important for the results and improved the validity and reliability of the experiment, given that we were interested in evaluating the use of an approach and tool by novice or non-expert analysts (Kitchenham, B. A. et al. 2002).

The experiment was also enhanced by the fact that the project was real, and the participants were actually real stakeholders of the system. Despite this, having only students as the represented stakeholders does place a limitation on the external validity of the experiment, given that in most real life project teams the requirements are elicited from different types of stakeholders with varied backgrounds and knowledge (Höst, Regnell & Wohlin 2000).

6.6 Cross Evaluation Discussion

Arguably the most important results and possible comparison is between the LAAS case study with 72 pieces of information in two hours with five participants and three information types, and the combined result of Groups 1 and 4 of the IUT experiment producing 79 pieced of information over the same time period, with the same number of participants, the same information types, and very similar projects. This outcome and the similarities between the procedures and conditions would suggest that the results obtained from the theoretical IUT experiment are consistent and comparable with the results achieved by the LAAS case study in practice. Although the LAAS case study participants had more experience and expertise in requirements elicitation than those from the IUT experiment, the results produced from the two evaluations were very similar. This would tend to indicate that the approach and tool managed to improve the performance of the more novice participants, and enabled them to achieve a level of performance on par with participants of greater experience and expertise. We can therefore say that the approach and tool combination has narrowed the gap between expert and novices requirements analysts.

One of the common observations across all three of the evaluations was that all the groups very quickly felt comfortable using the tool and confident in its direction of the requirements elicitation process. Specifically, there was no identified instance of participants trying or wanting to work against the advice of the tool. Leadership on the part of the analyst appeared to be in greater need and have a larger importance the less experienced the group was. In the LAAS case study with the most experienced group of participants, the analyst was actually more of a scribe than motivator, where as with the IUT experiment it was necessary for the analyst to encourage the other members. Likewise the less experienced the group was the more they appeared to depend on and utilize the advice offered by the tool for each of the required elicitation tasks.

From the results of the feedback questionnaires of all three evaluations, we can see that the MUSTER tool received high marks in the areas of understandability, learnability, and useability, demonstrating that the participants themselves found the

tool easy to understand, learn, and more importantly, use. Of the six main useability categories addressed in the feedback questionnaire, effectiveness consistently recorded the lowest marks across all three evaluations, although in all three cases this appeared to be a product of confusion with the statements in the questionnaire (i.e. insufficient explanations provided), rather than an accurate impression of the results produced. The attractiveness of the tool also scored relatively low mainly because of the bright and distinctive colours used for the interface, although this was intended as a positive aspect of the tool by enabling the user to more easily differentiate between the different sections of the main screen. The slightly higher scores for attractiveness from the LAAS case study maybe as a result of the participants being more experienced in requirements elicitation workshops, and therefore more concerned with the usefulness of the tool rather than its look and feel.

Throughout these evaluations a number of human factors could have had an impact on the internal validity of the process (Sadler & Kitchenham 1996). Because the project required the participants to adopt unfamiliar approaches and tools, this may have negatively biased the results. However this was counteracted by providing appropriate training to the participants before each of the evaluation sessions. Furthermore, the fact that the participants were not experts of the approach or tool was actually a desirable characteristic, as was the fact that all the participants were familiar with computer based tool and used them everyday in their normal work tasks, as is typical of most analysts and stakeholders involved in the design and development of software. The Hawthorne Effect (Landsberger 1958), where participants act differently as a result of being part of an investigation, rather than because of any approach or tool, may have positively biased the results by encouraging the participants to work harder than they normally would. However this too was counteracted by ensuring that the project did not attract any more attention during the evaluation phase than any other, and by attempting to make the researcher's involvement as unobtrusive and unobvious as possible.

It is also possible that experiment was positively influenced by expectation effects, whereby the participants may have been under the impression that the approach and tool they were about to use was better than existing ones, and therefore had the confidence that it would be an improvement without actually seeing it first. This

potential bias was also counteracted by ensuring that the selection of the participants for the project and therefore evaluation was not under the control or part of the evaluation process. As a result the deliberate selection (hand picking) of enthusiastic (positive predisposition) or cynical (negative predisposition) participants was not possible. In addition we made certain that the participants did not have a vested interest in the approach or the tool under evaluation, in either seeing it succeed or fail. This also counteracted possible intervention effects that may have biased the evaluations especially with respect to the feedback questionnaires were the participants in the evaluation were also the evaluators of the approach/tool.

6.7 Chapter Summary

It was as a result of the careful design that all the evaluations went according to plan with respect to the instruments and procedures used for the LAAS case study, INSA case study experiment, and IUT experiment. For all three of these evaluations we used both qualitative and quantitative metrics from multiple data sources (i.e. elicited information, observation sheet, and feedback questionnaire), which ensured the validity and reliability of our results. Using the same observation sheet and feedback questionnaire for all three evaluations, and a similar project for the both LAAS case study and IUT experiment further enhanced this.

This had the advantage of being able to compare the results from theory to practice, but also had the negative effect of reducing the coverage of the validation, i.e. only two system types. It was also important that the evaluation participants were a match with the intended audience and users of the approach and tool. We were able to maintain very high construct validity for all of the evaluations, despite the fact that the researcher was actively involved in the data collection and analysis. This potential bias was minimized by all results being checked by an independent party, who was a senior member of the research staff at LAAS-CNRS, and by the use of multiple data sources and analysis technique.

Through the evaluations presented in this chapter, we were able to firstly establish that collaborative group workshops are a useful technique for the early stages of requirements elicitation, especially when performed in conjunction with a situational approach and support tool. Furthermore, we were able to show that using the MUSTER tool improved the overall effectiveness, while the underlying OUTSET approach improved the overall efficiency, with respect to the elicitation of requirements for software systems. And although not all the findings were proven in practice with real stakeholders, the results from the experiments with students were consistent with the findings from the case study with actual practitioners, and therefore comparable.

CHAPTER 7: Conclusions

7.1 Summary Discussion

In **Chapter 1** of the thesis, we identified that requirements elicitation was not only a significant area of software engineering in need of further investigation, but that requirements elicitation is also one of the more critical and complex activities in the development of software systems. As a result, we proposed that the process of requirements elicitation could be improved in terms of efficiency, effectiveness, and useability by a collaborative situational approach to requirements elicitation with intelligent tool support, particularly during the early stages when performed by novice analysts in the absence of a defined methodology.

Through our review of the available literature on requirements elicitation presented in **Chapter 2**, we showed that there is a lack of appropriately flexible guidelines and sufficiently detailed steps which can be used by the majority of practitioners in typical projects. Although a large number of techniques, approaches, and tools currently exist for requirements elicitation, what is really needed are focused requirements elicitation approaches, which can be tailored dynamically depending on the situation. Such approaches would also require appropriate collaborative tool support that can combine multiple techniques into an integrated process which is both easy to use and useful to both analysts and stakeholders.

A survey of the current state of practice in requirements elicitation was described in **Chapter 3**, where it was seen that the general consensus in academia and industry was that significant improvement is needed. Furthermore, both experts and novices acknowledged the need to define a process for the early stages of requirements elicitation, and the importance of situational guidance and tool support. This finding not only confirmed a commonly held perception often found in the literature, but validated a major assumption on which the research was originally based.

The OUTSET approach presented in **Chapter 4** provides both researchers and practitioners of the software development process with an effective, efficient, useful, and useable requirements elicitation approach for this complex and critical activity. It is also expected that the integration of this systematic approach, and the resultant situation method from it, would lead to improvements throughout the entire software development process, as a direct result of improved requirements elicitation.

The MUSTER tool described in **Chapter 5** is both a Group Support System (GSS) for requirements elicitation, and a special purpose CASE/CAME application. In addition to embodying the OUTSET approach, MUSTER provides a number of benefits over existing requirements elicitation tools, in that it takes advantage of both collaborative requirements elicitation by being workshop centric, and the combination of multiple techniques within the process framework. Most importantly, MUSTER represents a tool for requirements elicitation that is not only useful, but one that does not require significant expertise or substantial experience on the part of the analysts and stakeholders to use.

In **Chapter 6** we presented the three empirical evaluations of the OUTSET approach and MUSTER tool, being the LAAS case study, INSA case study experiment, and IUT experiment. Through these evaluations, we were able to establish that collaborative group workshops are a useful technique for the early stages of requirements elicitation, and especially when performed in conjunction with a situational approach and support tool. We were also able to show that using the MUSTER tool improved the overall effectiveness, while the underlying OUTSET approach improved the overall efficiency, and the combination of the approach and tool provided a system that was both useful and useable. From the IUT experiment in particular, the MUSTER tool received more than three times the score of the manual tool for effectiveness, and more than double for usability. Just as important was the fact that the positive results from the experiment with students were comparable and therefore supported the positive results from the case study with practitioners.

Limitations of the Research

Not surprisingly, the empirical evaluations introduced the most evident limitations of the research. It is understandably difficult to completely separate the performance of

the approach and tool when they are evaluated in combination. Furthermore, any measurements used to determine the relative performance of an approach and tool combination will involve some trade-offs between the accuracy of the data collected, and the effort involved in its collection. An example of this in our research is the use of a feedback questionnaire to determine useability. Although such an instrument makes data collection and comparison relatively easy, it is largely subjective, and depends on the perceived useability of the approach and tool rather than the actual useability, which could have been measured using more direct but involved techniques. The same compromises have necessarily been made for the definitions and metrics we used to measure both efficiency and effectiveness.

Each of the research methods used, and research techniques employed, had their own shortcomings within the context of the research, and these have discussed in the relevant sections throughout the thesis. For example, we acknowledge that the feedback questionnaire used in the evaluations did not take into consideration, or weigh in any way, the potential differences in experience and expertise between the participating analysts. More broadly, and as a result of the evaluation methods chosen, there is some unavoidable limitation on the external validity of the research, given that the results may admittedly vary depending on not only the skill level of the analyst and stakeholders (i.e. the participants), but on the project type and environment as well (i.e. the context).

In general, a potential weakness of the research is that the presented approach and tool may be of little interest and benefit to expert practitioners of requirements elicitation. However this is generally outweighed by the fact that using the OUTSET approach and MUSTER tool presented in this research would likely provide benefit to the majority of practitioners and most software projects.

It may also be surmised that the success of requirements elicitation can only be determined when the quality of the results have been tested through the design, coding, and testing phases, and ultimately by the level of customer and user satisfaction upon completion. Although we have only evaluated the OUTSET approach and MUSTER tool at the end of the elicitation phase of software

development, the research still provides empirical evidence as to the improvements the approach and tool have made on the process.

It is not suggested that OUTSET and MUSTER should be the only approach and tool used for a complete requirements elicitation process in all situations, but rather they should be used where appropriate and in cooperation with other non-workshop based techniques such as observation and prototyping. Likewise, the output of the OUTSET and MUSTER is not intended to be the final requirements document, but instead a preliminary draft version to be used for subsequent iterations of elicitation, analysis, specification, and validation.

Strengths of the Research

The strengths of the research presented in this thesis includes the fact that both the state of the art and the state of practice were thoroughly investigated to confirm that requirements elicitation is not only a very worthwhile subject of enquiry, but one which was in need of attention. This subject was directly addressed by the design and construction of an innovative and original approach and tool, which was evaluated using both a case study and formal experiment, and was shown to be useable, effective, and efficient. Neither the OUTSET approach nor the MUSTER tool are confined to a specific software development process or methodology, and finally, both the approach and tool were specifically developed for the largest group of requirements elicitation practitioners (i.e. novice analysts) and the vast majority of software development projects (i.e. those without a defined requirements elicitation methodology).

7.2 Research Goals

Research Goal 1, which aimed to 'review and critically analyse the existing state of the art in requirements elicitation from the literature including the process, techniques, approaches, and tools', was thoroughly addressed in Chapter 2. In this part of the thesis we not only presented the definition, process, and scope of requirements elicitation for software systems (*Research Question 1*), but also described the relative strengths and weaknesses of the existing requirements elicitation techniques, approaches and tools (*Research Question 2*). As a result, we were able to identify from the literature several key components and features for a new and improved requirements elicitation approach and tool (*Research Question 3*) as presented in Chapters 4 and 5 respectively.

Research Goal 2, which aspired to 'investigate and survey the current state of practice in requirements elicitation as reported in the research literature, as well as from the perspective of both expert and novice analysts', was systematically addressed in Chapter 3. In this part of the thesis we examined the current state of practice in requirements elicitation not only from the perspective of the existing literature (*Research Question 4*), but also from experts in the field by using in-depth structured interviews (*Research Question 5*), and novices analyst through an online questionnaire (Research *Question 6*). Once again we were able to identify and subsequently implement from this survey several key components and features for a new and improved requirements elicitation approach and tool (*Research Question 7*) as in Chapters 4 and 5.

Research Goal 3, which endeavoured to 'design an approach and construct a tool to support novice analysts during the early stages of requirements elicitation for software development projects', was achieved by the results presented in Chapters 4 and 5. In Chapter 4 we described the OUTSET approach, which was successfully designed based on the identified key components from the results of the previous two research goals (*Research Question 8*). Likewise, in Chapter 5 we detailed the MUSTER tool, which was constructed from the same set of components to support, enhance, and extend the OUTSET approach (*Research Question 9*).

Finally, **Research Goal 4**, which sought to 'evaluate the performance of both the approach and tool for requirements elicitation in terms of improvements to efficiency, effectiveness, and useability', was successfully completed as can be seen in Chapter 6. In this chapter we presented a case study, case study experiment, and formal experiment, design and performed to evaluate the OUTSET approach and MUSTER tool when used for requirements elicitation by novice analyst, in terms of efficiency (*Research Question 10*), effectiveness (*Research Question 11*), and useability (*Research Question 12*).

From these results, obtained from satisfying the above goals, and answering the related questions, we can say that the research conducted and presented strongly supports both our general, and more importantly, our specific hypothesis that the OUTSET approach and MUSTER tool we have developed, as described and evaluated in this thesis, are not only useful and useable in terms of providing situational and group support, but also improve the efficiency and effectiveness of the early stages of requirements elicitation process when performed by novice analysis in the absence of a defined methodology.

7.3 Research Contributions

Contributions to the Body of Knowledge

The research presented in this thesis provides a number of important and valuable contributions to the body of knowledge as listed below:

1. A detailed review and critical analysis of the current state of the art in requirements elicitation, including the available process models, techniques, approaches, and tools.

2. A detailed survey and primary source of information on the current state of practice in requirements elicitation, including the trends and challenges of expert and novice analysts.

3. A new and improved approach specifically for requirements elicitation workshops called OUTSET, that utilizes, extends, and demonstrates a successful application of Situation Method Engineering (SME) principles.

4. A new and improved tool specifically for requirements elicitation workshops called MUSTER, that utilizes, extends, and demonstrates a successful application of Group Support System (GSS) principles, as well as embodying and enhancing the OUTSET approach.

5. Empirical evidence as to the performance of the OUTSET approach and MUSTER tool for requirements elicitation in both theory (experiments) and practice (case study) with respect to efficiency, effectiveness, and useability in particular.

Implications to Theory

The implications of this research on theory therefore include an increased general understanding of the field of requirements elicitation for software systems, both as a

result of the literature review and survey of practice. The approach and tool designed and constructed as part of the research provides a suitable foundation for other researchers to build on in order to develop additional support for requirements elicitation including domain specific processes. We have successfully demonstrated an example application of Situational Method Engineering (SME), Group Support Systems (GSS), and CASE/CAME tools specifically for requirements elicitation, and introduced the idea of using intelligent technologies to provide process guidance during the critical and complex activity.

Solutions to Problems in Practice

In addition, the research presented in this thesis offers a number of significant and helpful solutions to the problems in practice as listed below:

1. Provides practitioners with a new and improved approach and tool combination for requirements elicitation that is flexible, effective, efficient, useable, and useful, which can be readily applied to real-world projects.

2. Encourages and implements an appropriate degree of structure and rigor through process guidance, thereby substituting for a possible deficiency in experience and expertise of the participating novice analyst.

3. Directly addresses two areas of requirements elicitation workshops that were identified as lacking and needed, namely situational process guidance and group interaction support.

4. Directly addresses some of the issues in requirements elicitation that can lead to software project failure, by enabling stakeholder input early in the development process, and by encouraging a structured and rigorous approach.

5. By using the presented approach and tool, analysts will be better able to manage and reference large amounts of information, and have greater control over the process and the consistency of results.

Implications to Practice

Approaches and tools that improve the process of requirements elicitation have a number of important implications for practice. Because they ultimately endeavour to improve the overall performance and productivity of analysts in industry, approach and tool combinations such as OUTSET and MUSTER can help reduce the gap between expert and novice practitioners. The potential benefit from using such an approach and tool subsequently encourages their adoption by analysts, and therefore works towards reducing the gap between theory and practice.

The MUSTER tool is specifically intended to not only support novice analysts perform requirements elicitation, but to educate and train them on the process in practice, and ultimately to improve their performance and skill. The production of high quality requirements, as a result of practical experience using improved approaches and tools, enables a more accurate estimate of the time, effort, and cost required for software development projects. This is in addition to the potential for higher project success rates, and reductions in unexpected costs and the amount of expensive rework.

Novelty and Originality

The novelty and originality of the presented research includes the specific context that was investigated, being the support of novice analysts during the early stages of requirements elicitation for projects without a defined methodology. The combination of research methods was also quite special for this type of research, as a literature review was used to establish a sound theoretical foundation, a survey of practice to ground the research in the real world, and both a case study with multiple data sources and a formal experiment with cross validation were employed to evaluate and validate the work.

Unlike any other previous work, the OUTSET approach for requirements elicitation was developed using the principles of Situational Method Engineering (SME), which allowed different combinations of techniques to be used within a workshop environment with the required degree of structure and rigor. MUSTER is particularly

unique because it integrates the characteristics and functionalities of a CAME/CASE tool within a Group Support System (GSS), thereby interacts directly with the stakeholders and analysts, enabling them to work collaboratively on guiding the process and solving the problem.

The use of only Open Source technologies, and a plug-in architecture as a mechanism to store and transfer expert requirements elicitation knowledge was not only new and innovative, but also allowed us to easily implement soft computing and machine learning (i.e. intelligence) into the tool for process guidance and cognitive support.

7.4 Future Work

Although the research reported in this thesis provides both theory and practice with a number of valuable and much needed contributions, a number of avenues exist for future work in order to enhance the results and extend their implications.

The OUTSET approach could be further refined and expanded to include a more detailed procedure for project characterisation (i.e. by expanding the '3Ds' to include more situational characteristic such as the time and resources available for the project), and more specific construction rules (i.e. by providing comprehensive guidelines for the assembly and sequencing of process components). It would also be useful to develop a mechanism within the approach to allow and support changes in the characteristics of the project over its lifetime (e.g. the movement of a project from feasibility study to development, or from product evaluation to customization). Also, the creation of additional 'ready-made' methods for the approach (e.g. specifically for COTS evaluation and selection) would increase its applicability in terms of the number of project and system types it can immediately support.

Likewise, the MUSTER tool would benefit from the addition of more support plug-ins for specific system domains, such as embedded control systems and safety critical systems, and specific elicitation techniques, such as questionnaires, card sorting, and brainstorming. Further development of some of the partially implemented (e.g. Categoriser and Ask REG) and out of scope features (e.g. Discussion Boards and Workflow Engine) would provide MUSTER users with even greater flexibility and functionality in support of requirements elicitation, and the larger Requirements Engineering activity.

It would also be useful to explore the possibilities of reuse in the tool by extending the data repository to include a library of reusable domain objects and activities like clichés (Reubenstein & Waters 1991) or stereotypes (Cucchiarelli, Panti & Valenti 1994). Such a repository with reusable components could help speedup the requirements elicitation process, making the tool more attractive to both analysts and organisations because of potential savings in the time and effort required to elicit

requirements for each new project and system. The integration of the larger repository with other requirements tools such as DOORS (Telelogic 2005) and ArgoUML (Tigris 2005) could also provide some additional benefits, such as the ability to better manage the requirements change process, and enabling elicited requirements to be represented graphically.

With respect to the empirical evidence in support of the OUTSET approach and MUSTER tool, more case studies and formal experiments in different domains and situations is required to fully expose all the available functionality. During the presented evaluations, several features of both the approach and tool were not utilized at all, including those related more to project management (e.g. To Do and Open Issues lists), requirements management (e.g. Search and Processor engines), and much of the potential advice embedded in the support plug-ins.

During the evaluations, a number of potentially important factors were observed but not investigated, as these were deemed out of scope for the research. This included the impact of strong leadership and group dynamics on the relative performance of the evaluated groups. It would also be interesting to examine how the approach and tool could be used to overcome spatial and temporal barriers in cases such as distributed requirements elicitation for global software development, where multiple analysts and stakeholders may be located in different geographical locations, working in different time zones and calendars.

Reflections on the Research

Upon reflection, a number of aspects of the research can be identified that, if done differently, may have improved to some extent the outcomes achieved. More participants for both the expert interviews and novice questionnaire would naturally have provided more data on the state of practice in requirements elicitation, presenting a richer more complete picture of expert and novice attitudes and opinions. A second case study evaluation where the results from using the approach and tool could be tracked throughout the entire software development process may also have provided some additional insights. In hindsight, a more structured observation sheet could have made comparisons between the evaluations easier, and the feedback questionnaire might have benefited from more detailed explanations of certain terms

such as “good quality” with respect to the results produced by using the approach and tool. Despite these relatively minor modifications, overall the research project went very much to plan, and the results produced enabled the research goals and question to be addressed properly.

Appendix A: Invitation to Participate

Dear [*enter name of potential participant here*],

One of my PhD students (Chad Coulin) is conducting a survey for his doctoral research on the elicitation of requirements. Chad and I would be very grateful if you could find the time to assist him.

The research would involve either:
(A) Chad conducting a brief interview in person or over the phone based on a questionnaire, or
(B) You filling out the same questionnaire electronically and returning it to us via email.

Either way it should take no more than 1 hour of your time to complete.

If you are interested in participating in this research, please contact Chad Coulin via email to chadc@it.uts.edu.au or telephone on +61 (0) 402 411 284.

Best Wishes,
Didar Zowghi

Appendix B: Consent Form

I [*enter name of potential participant here*] agree to participate in the research project "A Collaborative and Combinational Approach to Requirements Elicitation with Process Guidelines and Tool Support [UTS HREC 2004-067A] being conducted by Chad Coulin of the University of Technology, Sydney [PO Box 123 Broadway NSW 2007 Australia, +61 (02) 9514 4446, chadc@it.uts.edu.au] for his degree of Doctor of Philosophy in Computing Sciences.

I understand that the purpose of this study will explore how and to what extent process guidelines and intelligent tool support can improve the process of requirements elicitation and the chances of success for software development projects and systems.

I understand that my participation in this research will involve either (A) a telephone or in person interview that may be audio-recorded with my permission and where I will receive a list of the questions prior to the interview, or (B) the completion of an electronic questionnaire to be distributed and returned via email, and either way should take no more than 60 minutes of my time to complete.

I am aware that I can contact Chad Coulin or his supervisor Didar Zowghi of the University of Technology, Sydney [PO Box 123 Broadway NSW 2007 Australia, +61 (02) 9514 1860, didar@it.uts.edu.au] if I have any concerns about the research. I also understand that I am free to withdraw my participation from this research project at any time I wish and without giving a reason.

I agree that Chad Coulin has answered all my questions fully and clearly regarding the research and my involvement in it.

I agree that the research data gathered from this project may be published in a form that does not identify me in any way.

Note:

This study has been approved by the University of Technology, Sydney Human Research Ethics Committee. If you have any complaints or reservations about any aspect of your participation in this research which you cannot resolve with the researcher, you may contact the Ethics Committee through the Research Ethics Officer, Ms Louise Abrams (ph: 02 9514 9615, Louise.Abrams@uts.edu.au) and quote the UTS HREC reference number. Any complaint you make will be treated in confidence and investigated fully and you will be informed of the outcome.

Appendix C: Expert Interview Questions

Introductory Briefing

Requirements elicitation can be broadly defined as the activities related to the acquisition of goals, constraints, and features for a proposed software based system by means of investigation, exploration, and analysis. Furthermore it is generally understood that requirements are elicited rather than captured or collected. This implies both a discovery and development element to the process.

The purpose of my doctoral research project is to develop a new collaborative (*multiple stakeholders working cooperatively with each other and the analyst*) and combinational (*multiple techniques and approaches used in combination where complementary*) approach to requirements elicitation with process guidelines and intelligent (*emulation of human perception, cognition, and reasoning*) tool support.

The focus and **context** of my doctoral research project is on supporting novice analysts performing the early stages of requirements elicitation in projects for software based information systems, in organizations that use and may even develop some software internally, but who's primary function and business is not the development of software or information technology.

The purpose of this questionnaire is to:

1. Confirm what is in the Requirements Engineering literature and what I believe from personal experience about the state of practice in requirements elicitation.

2. Act as a requirements elicitation session for the approach, guidelines, and tool I intend to develop.

1.0 General

No.	*Question*
1.1	What is your experience and expertise with respect to requirements elicitation (years in the software development industry, the number, type and size of the projects, etc.)?
1.2	What general comments could you make about the current state of practice in requirements elicitation from your personal experience?
1.3	What do you see as the major trends and challenges in requirements elicitation practice today?
1.4	Do you believe there is a need for new approaches to requirements elicitation, and if so, why? What could be the major elements of these new approaches?

2.0 Experts and Novices

No.	*Question*
2.1	How would you define an expert requirements elicitation analyst, and what are the major differences between them and novices?
2.2	In your opinion what are the most common pitfalls and mistakes made by novice analysts and why?
2.3	What types of support do you believe are needed to help novices perform requirements elicitation better and eventually become experts?

3.0 Process Guidelines

No.	*Question*
3.1	Do you believe it is important to develop a process for the early stages of requirements elicitation, and if so, why?
3.2	To what extent do you believe structure and rigor can be employed in the early stages of requirements elicitation, and if so, how?
3.3	How would you typically perform the process of requirements elicitation in the context described in terms of the activities performed and the techniques used?

4.0 Tool Support

No.	*Question*
4.1	What types of tools (including software) have you used for the elicitation of requirements and how effective have they been?
4.2	What features do you believe would be useful in a software tool to support the process of requirements elicitation in the context described?
4.3	What activities do you believe an intelligent software tool could support the analyst and stakeholders perform during requirements elicitation?

5.0 Evaluation

No.	*Question*
5.1	How would you define and measure the success of a requirements elicitation process?
5.2	In what ways do you believe a new approach to requirements elicitation could be evaluated and compared to existing approaches?
5.3	What metrics do you believe would be appropriate for the evaluation of a new approach to requirements elicitation?

6.0 Feedback

No.	*Question*
6.1	Do you believe the questionnaire was an appropriate length?
6.2	Do you believe the questions where sufficiently detailed?
6.3	In your opinion was any topic overlooked or not sufficiently addressed? If so, please specify.
6.4	Do you have any other general or specific comments about how this questionnaire could be improved?
6.5	Please add any other comments you feel may be useful for my research?

Appendix D: Online Novice Questionnaire

1. How many years experience do you have in projects eliciting requirements for software systems?
O 1

O 2

O 3

O 4

O Almost 5

2. How many projects have you been involved in the eliciting of requirements for software systems?
O 1-3

O 4-6

O 7-9

O 10-12

O More than 12

3. What is the typical duration in months of these projects that you have been involved in?
O Less than 6

O 6-12

O 13-24

O 25-36

O More than 36

4. What is the typical size of these projects in terms of the number of full time people involved in them?
O Less than 5

O 5-10

O 11-25

O 26-50

O More than 50

5. Which title best describes your typical role in these projects?
O Analyst

O Consultant

O Architect

O Designer

O Programmer

O Developer

O Software Engineer

O Systems Administrator

O Project Manager

6. How did you primarily learn to elicit requirements for software systems?

O Self taught (e.g. text books)

O Through experience

O University or TAFE course

O Professional training course

O On the job training

O From a mentor

O By observing others

7. How many of the projects you have been involved in have had problems related to poor requirements before and/or after the delivery of the software?

O None

O Few

O Some

O Most

O All

8. How would you rate your personal performance in eliciting requirements for the projects you have been involved in?

O Very poor

O Poor

O Fair

O Good

O Very good

9. What do you think is the overall level of performance in requirements elicitation by industry today?

O Very poor

O Poor

O Fair

O Good

O Very good

How often have you encountered the following issues during requirements elicitation?

	Never	Rarely	Some	Often	Always
10. Not all of the relevant stakeholders have been identified	O	O	O	O	O
11. Not all of the relevant stakeholders have been involved	O	O	O	O	O
12. The stakeholders do not know exactly what they want/need	O	O	O	O	O
13. The stakeholders can not clearly describe what they want/need	O	O	O	O	O
14. The stakeholders are not committed or cooperative	O	O	O	O	O
15. The stakeholders ask for unrealistic or non-feasible requirements	O	O	O	O	O
16. The requirements are not well understood	O	O	O	O	O
17. The requirements are not correctly documented	O	O	O	O	O
18. Requirements are missing or incomplete	O	O	O	O	O
19. Requirements are incorrect	O	O	O	O	O
20. Some requirements conflict with other requirements	O	O	O	O	O

21. What level of process guidance do you use for requirements elicitation in these projects?

O None (i.e. ad-hoc or random)

O Basic (i.e. high level overview)

O Defined (i.e. detailed steps and tasks)

How often do you use the following techniques for requirements elicitation?

	Never	Rarely	Some	Often	Always
22. Interviews	O	O	O	O	O
23. Questionnaires	O	O	O	O	O
24. Modeling (e.g. UML, DFDs, ERDs)	O	O	O	O	O
25. Scenarios (e.g. Use Cases, Customer Stories)	O	O	O	O	O
26. Viewpoint-oriented	O	O	O	O	O

27. Goal-directed	O	O	O	O	O
28. Prototypes	O	O	O	O	O
29. Brainstorming	O	O	O	O	O
30. Group Workshops	O	O	O	O	O
31. Observation	O	O	O	O	O
32. Apprenticing	O	O	O	O	O

How often do you use the following tools for the elicitation of requirements?

	Never	Rarely	Some	Often	Always
33. Word processors (e.g. Word)	O	O	O	O	O
34. Spreadsheets (e.g. Excel)	O	O	O	O	O
35. Databases (e.g. Access)	O	O	O	O	O
36. Document templates (e.g. IEEE)	O	O	O	O	O
37. Requirements Management tools (e.g. RequisitePro)	O	O	O	O	O
38. Diagramming tools (e.g. Visio)	O	O	O	O	O
39. Modeling tools (e.g. UML Toolkit)	O	O	O	O	O
40. White boards and flip charts	O	O	O	O	O
41. Audio and video recorders	O	O	O	O	O

How important are the following factors when you decide which requirements elicitation technique and/or tool to use?

	Not important at all	Not very important	Somewhat important	Important	Very important
42. Familiarity	O	O	O	O	O
43. Past success	O	O	O	O	O
44. Speed	O	O	O	O	O
45. Effort	O	O	O	O	O
46. Cost	O	O	O	O	O
47. Flexibility	O	O	O	O	O
48. Usefulness	O	O	O	O	O
49. Ease of use	O	O	O	O	O
50. The customer's preferences	O	O	O	O	O
51. Your organization's preferences	O	O	O	O	O

How often do you think the following types of assistance would help you improve your performance during requirements elicitation?

	Never	Rarely	Some	Often	Always
52. Process guidelines	O	O	O	O	O
53. Tool support	O	O	O	O	O
54. Increased education and training	O	O	O	O	O
55. Better elicitation techniques	O	O	O	O	O
56. More experience	O	O	O	O	O
57. Additional time	O	O	O	O	O
58. Bigger budget	O	O	O	O	O
59. Extra resources	O	O	O	O	O

How useful do you think tool support would be for the following requirements elicitation activities?

	Not useful at all	Not very useful	Somewhat useful	Useful	Very useful
60. Guidance through the process	O	O	O	O	O
61. Understanding the operating environment	O	O	O	O	O
62. Identifying stakeholders	O	O	O	O	O
63. Managing the communication with stakeholders	O	O	O	O	O
64. Recording and storing requirements	O	O	O	O	O
65. Managing and changing requirements	O	O	O	O	O
66. Elicitation techniques selection	O	O	O	O	O
67. Using different elicitation techniques	O	O	O	O	O
68. Managing and running interviews	O	O	O	O	O
69. Managing and running workshops	O	O	O	O	O

Appendix E: Requirements Specification

1. SYSTEM FEATURES AND FUNCTIONAL REQUIREMENTS

1.1 Login

This feature enables an authorized user to access the system functions.

1.1.1	The system shall allow the user to log in to the system.
1.1.2	The system shall verify the user's username and password upon login.
1.1.3	The system shall start the user's session upon successful login.
1.1.4	The system shall record the time of login for each user in an event log.
1.1.5	The system shall require the user to select a project.
1.1.6	The system shall require the user to select a language.

1.2 Logout

This feature enables a logged in user to exit the system completely.

1.2.1	The system shall allow the user to log out of the system.
1.2.2	The system shall end the user's session upon successful logout.
1.2.3	The system shall record the time of logout for each user in an event log.

1.3 Session Information

This feature provides useful and relevant information about the current project and session to the user.

1.3.1	The system shall display the name of the current project while in use.
1.3.2	The system shall display the current user while in use.
1.3.3	The system shall display the current date while in use.

1.4 System Information

This feature provides useful and relevant information about the system to the user.

1.4.1	The system shall provide contact details for technical support while in use.
1.4.2	The system shall display appropriate acknowledgements for the development of the OUTSET approach and MUSTER tool.
1.4.3	The system shall provide an internet link for additional information.

1.4.4	The system shall provide an email link for feedback.

1.5 Security Manager

This feature implements the necessary user security restrictions of the system.

1.5.1	The system shall only allow access to administration features to users that have the appropriate user type.
1.5.2	The system shall only allow access to projects to users that have the appropriate association.
1.5.3	The system shall only allow authorized users to access project information and system functions

1.6 Project Maintenance

This feature enables the user to maintain project information.

1.6.1	The system shall enable the user to view the existing projects in the system.
1.6.2	The system shall enable the user to add new projects to the system.
1.6.3	The system shall enable the user to modify information about existing projects in the system.
1.6.4	The system shall enable the user to delete existing projects in the system.
1.6.5	The system shall enable the user to define the type of requirements elicitation project, the type of system under investigation, and the type of deliverable required.
1.6.6	The system shall only allow administrators to access this feature.

1.7 User Maintenance

This feature enables the user to maintain user information.

1.7.1	The system shall enable the user to view the existing users in the system.
1.7.2	The system shall enable the user to add new users to the system.
1.7.3	The system shall enable the user to modify information about existing users in the system.
1.7.4	The system shall enable the user to delete existing users in the system.
1.7.5	The system shall enable the user to define the type of user, and the projects associated to that user.
1.7.6	The system shall only allow administrators to access this feature.

1.8 Activity Maintenance

This feature enables the user to maintain the standard set of activities prescribed by the system, as well as customize (tailor) the set of activities for a particular project.

1.8.1	The system shall enable the user to view the existing activities in the system.
1.8.2	The system shall enable the user to add new activities to the system.
1.8.3	The system shall enable the user to modify information about existing activities in the system.
1.8.4	The system shall enable the user to delete existing activities in the system.
1.8.5	The system shall enable the user to modify the activities for the current project, including sequence and description.
1.8.6	The system shall only allow administrators to access this feature.

1.9 Configuration Maintenance

This feature enables the user to change system configuration parameters.

1.9.1	The system shall enable the user to view the existing configuration parameters in the system.
1.9.2	The system shall enable the user to change the value of the existing configuration parameters in the system.
1.9.3	The system shall only allow administrators to access this feature.

1.10 Rules Maintenance

This feature enables the user to maintain rules and rule sets used when maintaining data of the information meta-model.

1.10.1	The system shall enable the user to view the existing rules and rule sets in the system.
1.10.2	The system shall enable the user to add new rules and rule sets to the system.
1.10.3	The system shall enable the user to modify information about existing rules and rule sets in the system.
1.10.4	The system shall enable the user to delete existing rules and rule sets in the system.
1.10.5	The system shall only allow administrators to access this feature.

1.11 Data Repository (System information and Meta-model instances)

This feature enables the storage and retrieval of all system and system related data and information.

1.11.1	The system shall enable the electronic storage and retrieval of system data.
1.11.2	The system shall enable the electronic storage and retrieval of meta-model data, including all information types.
1.11.3	The system shall enable the storage of information on relevant structures of the users' present work.
1.11.4	The system shall enable the storage of information for vision and design proposals.
1.11.5	The system shall enable the storage of overview information on technological options.
1.11.6	The system shall enable the storage of knowledge on the company strategy.
1.11.7	The system shall enable the storage of knowledge on the current organisation.
1.11.8	The system shall enable the storage of knowledge of government policy.
1.11.9	The system shall enable the storage and retrieval of all setup and configuration data including projects and users.
1.11.10	The system shall store information relevant to all the specific attributes for each type of system data and meta-model type.
1.11.11	The system shall enable the user to search the data repository for key words and phrases.
1.11.12	The system shall store the data for the information types separately.
1.11.13	The system shall be able to store and organize in a structured and presentable way a large amount of raw data from the elicitation process.

1.12 Information Maintenance (Population of the Data Meta-model)

This feature enables the user to maintain data in the repository related to the information meta-model.

1.12.1	The system shall enable and support the population of all information types in the meta-model.
1.12.2	The system shall allow the user to externalise and store their knowledge.
1.12.3	The system shall support the organisation of the stored information

1.12.4	The system shall support the traceability of requirements.
1.12.5	The system shall support the evolution of requirements
1.12.6	The system shall enable the user to view the existing data in the system.
1.12.7	The system shall enable the user to add new data to the system.
1.12.8	The system shall enable the user to modify the existing data in the system.
1.12.9	The system shall enable the user to delete existing data in the system.
1.12.10	The system shall provide templates for all the information types including attributes, operators, and rules.
1.12.11	The system shall enable the linking of one instance of an information type to other instances of other information types.
1.12.12	The system shall accept raw data (incorrect, incomplete, ambiguous, etc.).
1.12.13	The system shall accept data from multiple sources.
1.12.14	The system shall accept data from multiple users.

1.13 Process Guidance (Navigation Menu)

This feature provides the user with guidance on the process of requirements elicitation.

1.13.1	The system shall support facilitated meetings with predefined agendas and problem solving strategies.
1.13.2	The system shall enable the user to quickly and easily navigation from one task in the process to another.
1.13.3	The system shall enable the user to return to the start of the process or to any part of the system at any point in the process.
1.13.4	The system shall support articulation of the project concept.
1.13.5	The system shall support problem analysis.
1.13.6	The system shall support documentation of requirements.
1.13.7	The system shall support a systematic step by step approach.
1.13.8	The system shall provide automated support for the requirements elicitation process.
1.13.9	The system shall help identify and consult all likely sources of requirements
1.13.10	The system shall support users in analysing their own problems and identifying the need for change.
1.13.11	The system shall support the identification of stakeholders

1.13.12	The system shall support the management of the requirements management process.
1.13.13	The system shall support project planning.
1.13.14	The system shall support the management of knowledge development.
1.13.15	The system shall support the management of human communication
1.13.16	The system shall support identification of requirements.
1.13.17	The system shall support descriptions of typical users and user groups.
1.13.18	The system shall support descriptions of current work practices.
1.13.19	The system shall support identification of constraints.
1.13.20	The system shall support identification of acceptance criteria.
1.13.21	The system shall support identification of objective (organisation and stakeholders).
1.13.22	The system shall support identification of work practices to be supported and their functional requirements.
1.13.23	The system shall support identification and specification of quality attributes.
1.13.24	The system shall support identification and specification of requirements for user documentation, training, and user support.
1.13.25	The system shall support identification and description of human computer interface requirements.
1.13.26	The system shall encourage a complete specification to be written.

1.14 Task Execution

This feature enables the user to perform the tasks prescribed by the process guidance by using requirements elicitation techniques.

1.14.1	The system shall display relevant information about each task including description, sequence, recommended participants, and the types of information to be elicited.
1.14.2	The system shall provide the user with a number of specific techniques which can be used in order to perform each task.
1.14.3	The system shall enable the selection of different techniques from a list of those available to perform that task.
1.14.4	The system shall enable the user to quickly and easily enter data for the

	information types addressed by each task.
1.14.5	The system shall enable the user to update information about each task including general notes, and the estimated percentage complete.
1.14.6	The system shall provide the user with specific steps for each available technique which can be used in order to perform each task.

1.15 Technique Plug-ins

This feature provides a means for implementing requirements elicitation techniques into the system for use during workshops in conjunction with the process guidance.

1.15.1	The system shall provide the user with a number of different techniques which can be used to perform each task.
1.15.2	The system shall provide techniques that encourage intuition, imagination, collaboration, and common sense among workshop participants.
1.15.3	The system shall include a Questionnaire plug-in which provides the user with a series of relevant open probe questions in order to externalize and populate the required information for that task.
1.15.4	The system shall include an Interview plug-in which provides the user with a list of focused closed questions in order to externalize and populate the required information for that task.
1.15.5	The system shall include a Brainstorming plug-in which provides the user with a means for recording lots of ideas on a given subject very quickly, and then being able to organize and evaluate them.
1.15.6	The system shall include a Goal Oriented plug-in which provides the user with a template and instructions on how to use goal refinement and decomposition in order to externalize and populate the required information for that task.
1.15.7	The system shall include a Use Case plug-in which provides the user with a template and instructions on how to use Use Cases in order to externalize and populate the required information for that activity.
1.15.8	The system shall provide the user with guidance on interviewing users of the target system.
1.15.9	The system shall provide the user with guidance on the design and use of questionnaires.

1.16 Technique Selector

This feature provides support for the user in selecting which technique to use for each task prescribed by the process guidance during the process of requirements elicitation.

1.16.1	The system shall support the user in the selection of appropriate techniques for each activity.
1.16.2	The system shall take into account the skills of the current analyst and the current project situation when support the selection of techniques.

1.17 Glossary

This feature enables the user to maintain glossary items related to the project.

1.17.1	The system shall enable the user to view the existing terms
1.17.2	The system shall enable the user to add new terms
1.17.3	The system shall enable the user to modify information about existing terms
1.17.4	The system shall enable the user to delete existing terms

1.18 Data Dictionary

This feature enables the user to maintain data types related to the project.

1.18.1	The system shall enable the user to view the existing data types in the system for the project relevant to the target system.
1.18.2	The system shall enable the user to add new data types relevant to the target system for the project to the system.
1.18.3	The system shall enable the user to modify information about existing data types for the project in the system.
1.18.4	The system shall enable the user to delete existing data types for the project in the system.
1.18.5	The system shall enable the definition of a description, format, and allowed values for each data type.
1.18.6	The system shall enable the definition of relationships between data types.

1.19 References

This feature enables the user to maintain references to other material related to the project.

1.19.1	The system shall enable the user to view the existing references to other material related to the project in the system.
1.19.2	The system shall enable the user to add new references to other material related to the project to the system.
1.19.3	The system shall enable the user to modify information about existing references to other material related to the project in the system.
1.19.4	The system shall enable the user to delete existing references to other material related to the project in the system.

1.20 Appendixes

This feature enables the user to maintain appendixes necessary for the deliverable related to the project.

1.20.1	The system shall enable the user to view the existing appendixes necessary for the deliverable related to the project in the system.
1.20.2	The system shall enable the user to add new appendixes necessary for the deliverable related to the project to the system.
1.20.3	The system shall enable the user to modify information about existing appendixes necessary for the deliverable related to the project in the system.
1.20.4	The system shall enable the user to delete existing appendixes necessary for the deliverable related to the project in the system.

1.21 Issues List

This feature enables the user to maintain the issues list for the project.

1.21.1	The system shall enable the user to view the existing appendixes necessary for the deliverable related to the project in the system.
1.21.2	The system shall enable the user to add new appendixes necessary for the deliverable related to the project to the system.
1.21.3	The system shall enable the user to modify information about existing appendixes necessary for the deliverable related to the project in the system.
1.21.4	The system shall enable the user to delete existing appendixes necessary for the deliverable related to the project in the system.

1.22 Actions Lists

This feature enables the user to maintain the actions list for the project.

1.22.1	The system shall enable the user to view the existing actions related to the project in the system.
1.22.2	The system shall enable the user to add new actions related to the project to the system.
1.22.3	The system shall enable the user to modify information about existing actions related to the project in the system.
1.22.4	The system shall enable the user to delete existing actions related to the project in the system.

1.23 Ideas List

This feature enables the user to maintain the open ideas list and the related responses for the project.

1.23.1	The system shall enable the user to view the existing open ideas related to the project in the system.
1.23.2	The system shall enable the user to add new open ideas related to the project to the system.
1.23.3	The system shall enable the user to add new open idea responses related to the project to the system.
1.23.4	The system shall enable the user to modify information about existing open ideas related to the project in the system.
1.23.5	The system shall enable the user to delete existing open ideas related to the project in the system.
1.23.6	The system shall enable the user to add a response to an open idea anonymously in the system.

1.24 Reports and Exports

This feature provides the user with a number of useful reports, and a means of delivering the results of the requirements elicitation process and workshops.

1.24.1	The system shall provide the user with a standard list of reports on various aspects of the project and its progress.
1.24.2	The system shall provide both on-screen reports, and those that can be printed as document.

1.24.3	The system shall provide the user with a standard list of document formats in which to export the project information to. This should include at a minimum a Vision and Scope Statement, a Concept of Operations Document, and a Software Requirements Specification.
1.24.4	The system shall enable the user to export project information in predefined formats in the form of either HTML or MS Word documents.
1.24.5	The system shall provide reports that can be used during requirements walkthroughs with stakeholders.
1.24.6	The system shall provide reports that can be used for requirements inspections by stakeholders.

1.25 Resource Library

This feature provides the user with a number of useful resources to be used in support of the requirements elicitation process and workshops.

1.25.1	The system shall provide the user with a variety of resources relevant and useful during the requirements elicitation process. This should include appropriate templates, checklists, and examples.
1.25.2	The system shall enable the user to download local versions of all available resources to be used offline.

1.26 Online Help

This feature provides the user with assistance on using the systems, and using workshops for requirements elicitation.

1.26.1	The system shall provide the user with online help relevant to the system including a getting started guide and a user manual.
1.26.2	The system shall provide the user with online help relevant to the process of requirements elicitation.
1.26.3	The system shall provide the user with online help relevant to preparing and performing workshops.
1.26.4	The system shall enable the user to search the help by key word of phrase.

1.27 Data Search

This feature enables the user to search the data stored in the repository related to the information meta-model.

1.27.1	The system shall enable the user to search the data populated in the meta-model by key word or phrase.
1.27.2	The system shall enable the user to specify which information types should be included when searching the data in populated in the meta-model.
1.27.3	The system shall display to the user all the relevant information of the results from the search on the screen.

1.28 Session Management

This feature enables the user to maintain session information relevant to the project.

1.28.1	The system shall enable the user to view the existing sessions related to the project in the system.
1.28.2	The system shall enable the user to add new sessions related to the project to the system.
1.28.3	The system shall enable the user to modify information about existing sessions related to the project in the system.
1.28.4	The system shall enable the user to delete existing sessions related to the project in the system.
1.28.5	The system shall enable the storage and retrieval of all information related to sessions including the dates and time, participants, and location.

1.29 Event Log

This feature provides a historical record of all events and actions performed by the users and the system itself relevant to each project.

1.29.1	The system shall record in an electronic format all events and actions performed by the users and the system. This includes events such as login.
1.29.2	The system shall record all the relevant information about each event including at a minimum.
1.19.3	The system shall record event information in a way that can be searched.

2. EXTERNAL INTERFACE REQUIREMENTS

2.1	The system shall accept input from standard computer devices including keyboard and mouse.
2.2	The system shall allow output to standard computer printing devices.

3. PERFORMANCE REQUIREMENTS

3.1	The system shall support a maximum load of 1000 concurrent users.
3.2	The system shall support a maximum rate of 100 transactions per second.
3.3	The system shall support a maximum amount of 1 gigabyte of data.

4. DESIGN CONSTRAINTS

4.1	The system shall be web-based, i.e. the user interface will be accessed via a standard web browser.
4.2	The system shall be developed using only free technologies.
4.3	The system shall run on standard computer hardware platforms, operating systems.
4.4	The system server side shall run on the Apache web server.

5. SOFTWARE SYSTEM ATTRIBUTES

5.1	The system shall have a user interface that is useful, easy to use, and simple to understand for both analysts and stakeholders.
5.2	The system shall comply with standard internet (web page) conventions for navigation and operation.
5.3	The system shall have be available 95% of the time.
5.4	The system shall be completely recoverable in 4 hours.
5.5	The system shall be developed in such a way and using such technologies, techniques and tools, that is can be easy maintained by novice developers.
5.6	The system shall support geographically distributed users.
5.7	The system shall facilitate the input of elicited information as much as possible by reducing the amount of typing required by the user, and increasing the speed at which information can be entered wherever possible.
5.8	The system shall be attractive to the users, in that it should want to be used

	by the users for requirements elicitation, and it should not alienate or intrude upon them.

6. OTHER REQUIREMENTS

6.1	The system shall involve both the customers and the users of the target system during the requirements elicitation process.
6.2	The system shall support the analyst in maintaining good relationships between the stakeholders of the target system including open communication and access to information related to the project.
6.3	The system shall support the development of a shared meeting of the target system being specified.
6.4	The system shall support communication between people from geographically distributed locations and diverse backgrounds.
6.5	The system shall support multiple (unlimited) iterations of the requirements elicitation process within a single project.
6.6	The system shall support the incremental elicitation of requirements information across multiple sessions within a single project.

Appendix F: Feedback Questionnaire

Please answer all of the following questions.

Group: Date: Place a cross ("X") in the column that most represents your position for each of the statements below.	Strongly agree	Agree	Neutral	Disagree	Strongly disagree
The tool was easy to understand					
The information was presented and organized in an way and format that is easy to understand					
The tool was not overly complex					
The tool was easy to learn how to use					
It was easy to remember how to do things in the tool					
The tool had a logical structure and made sense to me					
The user interface of the tool was attractive					
The layout and language of the tool was consistent					
The layout and language of the tool was convenient					
The user interface of the tool was easy to navigate and move around					
It was easy to find what I was looking for in the tool					
The tool was easy to use					
The tool was efficient to use					
The tool helped me be productive					
The response time of the tool was good					
The tool was effective to use					
The results of the tool were useful					
The results of the tool were of a good quality					
The tool was flexible enough to be used in other system development projects					

The tool has the necessary functions and features to support the task of requirements elicitation					
I felt comfortable using the tool					
I would use the tool again					
The tool was better than others I have used					
I would recommend the tool to others					
Overall I am satisfied with the tool					
Using the tool save me time					
Using the tool save me effort					
Using the tool would improved the quality of the results					
My knowledge of requirements elicitation has increased by using the tool					

List the three most positive aspects of the tool, such as what was useful:

1.

2.

3.

List the three most negative aspects of the tool, such as what could be improved:

1.

2.

3.

Please check all your answers before submitting the questionnaire.

Appendix G: Observation Sheet

Group:	Date:

1. Explain if the participants appeared to understand what was required of them for the session, and if this changed over time. Identify what parts they did not seem to understand.

2. Explain if the participants appeared to find the tool easy or difficult to use through out the session, and if this changed over time. Identify what parts seemed particularly easy or difficult for the participants.

3. Explain if the participants appeared to find the tool helpful or unhelpful throughout the session, and if this changed over time. Identify what parts seemed particularly helpful or unhelpful for the participants.

4. Explain how the participants used the advice offered by the tool to elicit information throughout the session, such as how much they referred to it and how much it determined their actions.

5. Explain briefly the interaction between the participants throughout the session, such as how much each member contributed, and what points were discussed in detail and by how many of the participants.

Please use the other side of the page if more space is required.

References

Agarwal, R & Tanniru, MR 1990, 'Knowledge Acquisition Using Structured Interviewing: An Empirical Investigation', *Journal of Management Information Systems*, vol. 7, no. 1, pp. 123-40.

Akao, Y 1995, *Quality Function Deployment: Integrating Customer Requirements into Product Design*, Productivity Press, Cambridge, USA.

Alderson, A 1991, 'Meta-case Technology', European Symposium on Software Development Environments and CASE Technology, Konigswinter, Germany, June 17-19.

Alexander, I 2007, *Requirements Engineering Tools and Vendors*, 2007, <http://easyweb.easynet.co.uk/~iany/other/vendors.htm>.

Alexander, IF 2005, 'A Taxonomy of Stakeholders: Human Roles in System Development', *International Journal of Technology & Human Interaction*, vol. 1, no. 1, pp. 23-59.

Alexander, IF & Maiden, N 2004, *Scenarios, Stories, Use Cases - Through the Systems Development Life-Cycle*, Wiley, UK.

Alho, K & Sulonen, R 1998, 'Supporting Virtual Software Projects on the Web', Seventh IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE '98), Stanford, USA, June 17-19.

American Society for Quality 2006, *Glossary*, <http://www.asq.org/glossary/index.html>.

Andreou, AS 2003, 'Promoting software quality through a human, social and organisational requirements elicitation process', *Requirements Engineering*, vol. 8, pp. 85-101.

Arthur Andersen 1988, *Method/1 User Manual*, Arthur Andersen & Co.

Atlantic Systems Guild 2003, *Volere Requirements Specification Template*.

---- 2007, *Requirements Tools*, 2007, <http://www.volere.co.uk/tools.htm>.

August, JH 1991, *Joint Application Design: The Group Session Approach to Systems Design*, Prentice-Hall, Englewood Cliffs, USA.

Ball, LJ & Ormerod, TC 2000, 'Putting ethnography to work: the case for a cognitive ethnography of design', *International Journal of Human–Computer Studies*, vol. 53, no. 1, p. 147–68.

Basili, V & Turner, A 1975, 'Iterative Enhancement: A Practical Technique for Software Development', *IEEE Transactions on Software Engineering*, vol. 1, no. 4, pp. 390-6.

Beck, K & Cunningham, W 1989, 'A Laboratory For Teaching Object-Oriented Thinking', Object Oriented Programming Systems Languages and Applications, New Orleans, USA, October 1-6.

Berenbach, BA 2004, 'Practitioner reports: Comparison of UML and text based requirements engineering', 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications, Vancouver, Canada, October 24-28.

Bevan, N 2006, 'Practical Issues in Usability Measurement', *Interactions*, vol. 13, no. 6, pp. 42-3.

Beyer, H & Holtzblatt, K 1997, *Contextual Design : A Customer-Centered Approach to Systems Designs*, Morgan Kaufmann, USA.

Beyer, HR & Holtzblatt, K 1995, 'Apprenticing with the customer', *Communications of the ACM*, vol. 38, no. 5, pp. 45-52.

Boehm, B 1981, *Software Engineering Economics*, Prentice-Hall, New Jersey, USA.

Boehm, B, Bose, P, Horowitz, E & Lee, M 1994, 'Software Requirements As Negotiated Win Conditions', First International Conference on Requirements Engineering, Colorado Springs, USA, April 18-22.

Boehm, B, Egyed, A, Kwan, J, Port, D, Shah, A & Madachy, R 1998, 'Using the WinWin Spiral Model: A Case Study', *IEEE Computer*, vol. 31, no. 7, pp. 33-44.

Boehm, BW 1988, 'A Spiral Model of Software Development and Enhancement', *Computer*, vol. 21, no. 5, pp. 61-88.

Borland 2005, *CaliberRM*, <http://www.borland.com/us/products/caliber/>.

Bostrum, RP 1989, 'Successful application of communication techniques to improve the systems development process', *Information and Management*, vol. 16, no. 5, p. 279–95.

Braun, A, Bruegge, B & Dutoit, A 2001, 'Supporting Informal Meetings in Requirements Engineering', 7th International Workshop on Requirements Engineering for Software Quality, Interlaken, Switzerland, June 4-5.

Briggs, B & Grünbacher, P 2002, 'EasyWinWin: Managing Complexity in Requirements Negotiation with GSS', 35th Hawaii International Conference on System Sciences, Big Island, Hawaii, January 7-10.

Brinkkemper, S 1996, 'Method engineering: engineering of information systems development methods and tools', *Information and Software Technology*, vol. 38, pp. 275-80.

Brinkkemper, S, Lyytinen, K & Welke, R (eds) 1996, *Method Engineering : Principles of method construction and tool support*, Springer.

Brinkkemper, S, Saeki, M & Harmsen, F 1999, 'Meta-Modelling Based Assembly Techniques for Situational Method Engineering', *Information Systems*, vol. 24, no. 3, pp. 209-28.

Brooke, J 1996, 'SUS: A "quick and dirty" usability scale.' in PW Jordan, B Thomas, BA Weerdmeester & AL McClelland (eds), *Usability Evaluation in Industry*, Taylor and Francis, London, pp. 189-94.

Brooks, FP 1987, 'No Silver Bullet: Essence and Accidents of Software Engineering', *IEEE Computer*, vol. 20, no. 4, pp. 10-9.

Bubenko, JA & Wangler, B 1993, 'Objectives driven capture of business rules and of information systems requirements', International Conference on Systems, Man and Cybernetics, Le Touquet, France, October 17-20.

Buren, JV & Cook, DA 1998, *Experiences in the Adoption of Requirements Engineering Technologies*, Software Technology Support Center.

Burrell, G & Morgan, G 1979, *Sociological Paradigms and Organizational Analysis*, Heinemann.

Carlshamre, P & Karlsson, J 1996, 'A usability-oriented approach to requirements engineering', Second International Conference on Requirements Engineering, Colorado Springs, USA, April 15-18.

Carmel, E, Whitaker, RD & George, JF 1993, 'PD and joint application design: a transatlantic comparison', *Communications of the ACM*, vol. 36, no. 6, p. 40–8.

Cediti 2005, *Objectiver*, <http://www.objectiver.com/en/home/>.

Centra 2005, *Centra Live for eMeetings*, <http://www.centra.com/>.

Chang, CF, Krishna, A & Ghose, AK 2003, 'Agent-assisted Distributed Requirements Elicitation And Management', Fifteenth International Conference on Software Engineering & Knowledge Engineering, San Franciso, USA, July 1-3.

Chatzoglou, PD 1997, 'Factors affecting completion of the requirements capture stage of projects with different characteristics', *Information and Software Technology*, vol. 39, pp. 627-40.

Checkland, P & Scholes, J 1990, *Soft Systems Methodology in Action*, John Wiley & Sons, New York, USA.

Chin, G, Rosson, MB & Carroll, JM 1997, 'Participatory Analysis: Shared Development of Requirements from Scenarios', Human Factors in Computing Systems Conference, Atlanta, USA, March 22-27.

Christel, MG & Kang, KC 1992, *Issues in Requirements Elicitation*, CMU/SEI-92-TR-012, Carnegie Mellon University - Software Engineering Institute.

Christel, MG, Wood, DP & Stevens, SM 1993, *AMORE: The Advanced Multimedia Organizer for Requirements Elicitation*, Technical Report CMU/SEI-93-TR-12 ESC-TR-93-189, Carnegie Mellon University - Software Engineering Institute.

Chung, L, Nixon, BA, Yu, E & Mylopoulos, J 1999, *Non-functional Requirements in Software Engineering*, Kluwer Academic Publishers, USA.

City University 2005, *ART-SCENE*, <http://www-hcid.soi.city.ac.uk/research/Artsceneindex.html>.

Cockburn, A 2001, *Writing Effective Use Cases*, Addison-Wesley, Reading, USA.

Compendium Institute 2005, *Compendium*, <http://www.compendiuminstitute.org/tools/compendium.htm>.

Constantine, L & Lockwood, LAD 1999, *Software for Use: A practical guide to the models and methods of usage-centered design*, Addison-Wesley, Reading, USA.

Corel 2005, *iGrafx FlowCharter*, <http://www.igrafx.com/products/flowcharter/>.

Coulin, C & Zowghi, D 2005a, 'Requirements Elicitation for Complex Systems: Theory and Practice', in JL Maté & A Silva (eds), *Requirements Elicitation in Requirements Engineering for Socio-Technical Systems*, Idea Group, USA.

---- 2005b, 'What Do Experts Think About Elicitation? - A State of Practice Survey', Australian Workshop on Requirements Engineering, Melbourne, Australia, November 22.

Coulin, C, Zowghi, D & Sahraoui, A 2005, 'A Workshop-Centric Situational Approach for Requirements Elicitation', International Workshop on Situational Requirements Engineering Processes, Paris, France, August 29-30.

---- 2006, 'A Situational Method Engineering Approach to Requirements Elicitation Workshops in the Software Development Process', *Software Process: Improvement and Practice*, vol. 11, no. 5, pp. 451-64.

Creative Research Systems 2005, *The Survey System*, <http://www.surveysystem.com>.

CREWS 1999, *Cooperative Requirements Engineering with Scenarios*, 2007, <http://www-i5.informatik.rwth-aachn.de/i5new/projects/crews/>.

Cucchiarelli, A, Panti, M & Valenti, S 1994, 'Supporting User-Analyst Interaction in Functional Requirements Elicitation', First Asia-Pacific Software Engineering Conference, Tokyo, Japan, December 7-9.

Cybulski, JL 1999, *Tool Support for Requirements Engineering*, Department of Information Systems, University of Melbourne.

---- 2002, 'Automatic Refinement of User Requirements: A Case Study in Software Tool Evaluation', Thirteenth Australasian Conference on Information Systems, Melbourne, Australia, December 4-6.

Czuchry, AJ & Harris, DR 1988, 'KBRA: A New Paradigm for Requirements Engineering', *IEEE Expert*, vol. 3, no. 4, pp. 21-35.

Dahanayake, ANW 1998, 'Evaluation of the Strength of Computer Aided Method Engineering for Product Development Process Modeling', 9th International Conference on Database and Expert Systems Applications, Vienna, Austria, August 24-28.

Dardenne, A, van Lamsweerde, A & Fickas, S 1993, 'Goal-Directed Requirements Acquisition', *Science of Computer Programming*, vol. 20, no. 1-2, pp. 3-50.

Davis, AM 1990, *Software Requirements: Analysis and Specification*, Prentice-Hall, New Jersey, USA.

---- 1993, *Software Requirements: Objects, Functions, & States*, Revision edn, Prentice Hall PTR, Upper Saddle River, USA.

---- 2004, *Just Enough Requirements Management: Where Marketing and Development Meet*, Dorset House, New York, USA.

Davis, AM & Hickey, AM 2002, 'Requirements Researchers: Do We Practice What We Preach?' *Requirements Engineering Journal*, vol. 7, no. 2, pp. 107-11.

Davis, GB 1982, 'Strategies for information requirements determination', *IBM Systems Journal*, vol. 21, no. 1, pp. 4-30.

de Bono, E 1999, *Six Thinking Hats*, Back Bay Books, USA.

de Freitas, RM, Borges, MRS, Santoro, FM & Pino, JA 2003, 'Groupware Support for Cooperative Process Elicitation', 9th International Workshop on Groupware, Grenoble, France, September 28 - October 2.

Delbecq, AL, Van de Ven, AH & Gustafson, DH 1975, *Group Techniques for Program Planning: A Guide to Nominal Group and Delphi Processes*, Scott, Foresman and Company, Glenview, USA.

DeMarco, T & Plauger, PJ 1979, *Structured Analysis and System Specification*, Prentice Hall, New York, USA.

den Hengst, M, van de Kar, E & Appelman, J 2004, 'Designing Mobile Information Services: User Requirements Elicitation with GSS Design and Application of a Repeatable Process', 37th Hawaii International Conference on System Sciences, Big Island, Hawaii, January 5-8.

Dey, I 1999, *Grounding Grounded Theory: Guidelines for Qualitative Inquiry*, Academic Press, USA.

DSDM Consortium 2006, *Dynamic Systems Development Method*, <http://www.dsdm.org>.

Duggan, EW & Thachenkary, CS 2004, 'Integrating nominal group technique and joint application development for improved systems requirements determination', *Information and Management*, vol. 41, no. 4, pp. 399-411.

Durán Toro, A, Bernárdez Jiménez, B, Ruiz Cortés, A & Toro Bonilla, M 1999, 'A Requirements Elicitation Approach Based in Templates and Patterns', Workshop em Engenharia de Requisitos, Buenos Aires, Argentina, September 9-10.

Edwards, ML, Flanzer, M, Terry, M & Landa, J 1995, 'RECAP: A Requirements Elicitation, Capture and Analysis Process Prototype Tool for Large Complex Systems', IEEE International Conference on Engineering of Complex Computer Systems, Ft. Lauderdale, USA, November 6-10.

El Emam, K & Madhavji, NH 1996, 'An instrument for measuring the success of the requirements engineering process in information systems development', *Empirical Software Engineering*, vol. 1, no. 3, pp. 201-40.

Ellis, CA, Gibbs, SJ & Rein, GL 1991, 'Groupware: Some issues and experiences', *Communications of the ACM*, vol. 34, no. 1, pp. 38-58.

Falafel Software 2005, *Active! Focus*, <http://www.falafel.com/Order/ActiveFocus/tabid/101/Default.aspx>.

Farlex 2006, *The Free Dictionary*, <http://www.thefreedictionary.com>.

Feblowitz, M, Greenspan, S, Reubenstein, H & Walford, R 1996, 'ACME/PRIME: Requirements Acquisition for Process-Driven Systems', Eighth International Workshop on Software Specification and Design, Paderborn, Germany, March 22-23.

Fetterman, DM 1997, *Ethnography: Step-by-Step*, 2nd edn, Applied Social Research Methods, Sage, USA.

Finkelstein, A 1994, 'Requirements Engineering: a review and research agenda', Asia Pacific Conference on Software Engineering, Tokyo, Japan, December 7-9.

Finkelstein, A & Kramer, J 2000, 'Software Engineering: A Roadmap', The Future of Software Engineering, Limerick, Ireland, June 4-11.

Firesmith, D & Henderson-Sellers, B 2001, *The OPEN Process Framework: An Introduction*, Addison-Wesley, USA.

Foddy, W 1994, *Constructing Questions for Interviews and Questionnaires*, Cambridge University Press, Cambridge, UK.

Fowler, FJ 1995, *Improving Survey Questions: Design and Evaluation*, vol. 38, Applied Social Research Methods, SAGE Publications, Thousand Oaks, USA.

Fowler, M 2005, *The New Methodology*, <www.martinfowler.com>.

Frøkjær, E, Hertzum, M & Hornbæk, K 2000, 'Measuring Usability: Are Effectiveness, Efficiency, and Satisfaction Really Correlated?' CHI 2000 Conference on Human Factors in Computing Systems, The Hague, The Netherlands, April 1-6.

Gaines, BR & Shaw, MLG 1995, 'Concept Maps as Hypermedia Components', *International Journal of Human-Computer Studies*, vol. 43, no. 3, pp. 323-61.

Gambhir, SG 2001, 'An Investigation of Facilitator-Assisted and CONOPS-Based Requirements Elicitation Methods Using a 2 x 2 Factorial Experiment Design', *Systems Engineering*, vol. 4, no. 4, pp. 272-86.

Gause, DC & Weinberg, GM 1989, *Exploring Requirements: Quality Before Design*, Dorset House, New York, USA.

George Mason University 2005, *InSERT Dictionary*, 2005, <http://www.gmu.edu/departments/seor/insert/dictionary/Default.htm>.

Gilb, T 1988, *Principles of Software Engineering Management*, Addison-Wesley Professional, USA.

Goguen, JA 1996, 'Formality and Informality in Requirements Engineering', IEEE International Conference on Requirements Engineering, Colorado Springs, USA, April 15-18.

Goguen, JA & Linde, C 1993, 'Techniques for Requirements Elicitation', IEEE International Symposium on Requirements Engineering, San Diego, CA, USA, January 4-6.

Goldin, L & Berry, DM 1994, 'AbstFinder, A Prototype Natural Language Text Abstraction Finder for Use in Requirements Elicitation', *Automated Software Engineering*, vol. 4, no. 4, pp. 375-412.

Gottesdiener, E 2001, 'Collaborate for Quality', *Software Testing and Quality Engineering Magazine*, pp. 1-8.

---- 2002, *Requirements by Collaboration: Workshops for Defining Needs*, Addison-Wesley, Boston, USA.

Graham, I 1998, *Requirements Engineering and Rapid Development: An Object-Oriented Approach*, Addison-Wesley, Great Britain.

GroupSystems 2005, *GroupSystems*, <http://www.groupsystems.com>.

Gulliksen, J, Lantz, A & Boivie, I 1999, 'User Centered Design – Problems and Possibilities', *SIGCHI Bulletin*, vol. 31, no. 2, pp. 25-35.

Hamel, J, Dufour, S & Fortin, D 1993, *Case study methods*, Sage, Newbury Park, USA.

Hands, K, Peiris, DR & Gregor, P 2004, 'Development of a computer-based interviewing tool to enhance the requirements gathering process', *Requirements Engineering*, vol. 9, pp. 204-16.

Hannola, L, Elfvengren, K & Tuominen, M 2005, 'Improving Requirements Elicitation with GSS in Software Development', Annual ISPIM Conference, International Society for Professional Innovation Management, Porto, Portugal, June 19-22.

Hassan, S & Salim, SS 2004, 'A Tool to Support Collaborative Requirements Elicitation using Viewpoint Approach', 1st International Conference on Informatics, Izmir, Turkey, September 1-4.

Haumer, P, Pohl, K & Weidenhaupt, K 1998, 'Requirements Elicitation and Validation with Real World Scenes', *IEEE Transactions on Software Engineering*, vol. 24, no. 12, pp. 1036-54.

Helmer-Hirschberg, O 1967, *Analysis of the Future: The Delphi Method*, P-3558, RAND Corporation.

Henderson-Sellers, B 2002, 'Process Metamodelling and Process Construction: Examples Using the OPEN Process Framework (OPF)', *Annals of Software Engineering*, vol. 14, p. 341–62.

Herela, D & Greenberg, S 1998, 'Using a Groupware Space for Distributed Requirements Engineering', Seventh Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, Stanford, USA, June 17-19.

Hickey, AM 2003, 'Requirements Elicitation Techniques: Analyzing the Gap between Technology Availability and Technology Use', *Comparative Technology Transfer and Society*, vol. 1, no. 3, pp. 279-302.

Hickey, AM & Davis, AM 2002, 'The Role of Requirements Elicitation Techniques in Achieving Software Quality', Eighth International Workshop of Requirements

Engineering: Foundation for Software Quality, Essen, Germany, September 9-10.

---- 2003a, 'Barriers to Transferring Requirements Elicitation Techniques to Practice', Sixth International Conference on Business Information Systems, June.

---- 2003b, 'Elicitation Technique Selection: How Do Experts Do It?' Eleventh IEEE International Requirements Engineering Conference, Monterey Bay, USA, September 8-12.

---- 2003c, 'Requirements Elicitation and Elicitation Technique Selection: A Model for Two Knowledge-Intensive Software Development Processes', 36th Annual Hawaii International Conference on System Sciences, Big Island, Hawaii, January 6-9.

---- 2004, 'A Unified Model of Requirements Elicitation', *Journal of Management Information Systems*, vol. 20, no. 4, pp. 65-84.

Hickey, AM, Dean, DL & Nunamaker, JF 1999, 'Establishing a Foundation for Collaborative Scenario Elicitation', *The DATA BASE for Advances in Information Systems*, vol. 30, no. 3-4, pp. 92-110.

Hinkle, D 1965, 'The change of personal constructs from the viewpoint of a theory of implications', Doctoral Dissertation thesis, Ohio State University.

Hoffer, JA, George, JF & Valacich, JS 2002, *Modern Systems Analysis and Design*, Third edn, Prentice Hall, USA.

Hofmann, HF & Lehner, F 2001, 'Requirements Engineering as a Success Factor in Software Projects', *IEEE Software*, vol. 18, no. 4, pp. 58-66.

Holmes, JW 2004, *Unit Command Climate Assessment and Survey System (UCCASS) v1.8.0*, <http://www.bigredspark.com/survey.html>.

Holtzblatt, K & Beyer, HR 1995, 'Requirements Gathering: The Human Factor', *Communications of ACM*, vol. 38, no. 5, pp. 30-2.

Höst, M, Regnell, B & Wohlin, C 2000, 'Using Students as Subjects - A Comparative Study of Students and Professionals in Lead-Time Impact Assessment', *Empirical Software Engineering*, vol. 5, p. 201–14.

IBM 2005, *Rational RequisitePro*, <http://www-306.ibm.com/software/awdtools/reqpro/>.

---- 2006, *Rational Rose*, <http://www-306.ibm.com/software/awdtools/developer/rose/>.

IEEE 1990, *IEEE Std 610.12-1990 Standard Glossary of Software Engineering Terminology*.

---- 1998a, *IEEE Std 830-1998 Recommended Practice for Software Requirements Specifications*.

---- 1998b, *IEEE Std 1362 System Definition - Concept of Operations (ConOps) Document*.

---- 2004, *SWEBOK: Guide to the Software Engineering Body of Knowledge*.

INCOSE 2007, *INCOSE Requirements Management Tools Survey*, 2007, <http://www.paper-review.com/tools/rms/read.php>.

INSA Toulouse 2006, *National Institute of Applied Sciences, Toulouse*, <http://www.insa-toulouse.fr>.

ISO 1998, *ISO 9241-11:1998 Ergonomic requirements for office work with visual display terminals (VDTs) -- Part 11: Guidance on usability*.

IUT Blagnac 2006, *University Institute of Technology, Blagnac*, <http://www.iut-blagnac.fr>.

Jackson, M 1995, 'The World and the Machine', Seventeenth IEEE International Conference on Software Engineering, Seattle, USA, April 24-28.

---- 2000, *Problem Frames: Analyzing and Structuring Software Development Problems*, Addison-Wesley, Boston, USA.

Jarzabek, S & Huang, R 1998, 'The Case for User-Centered CASE Tools', *Communications of the ACM*, vol. 41, no. 8, pp. 93-9.

Kaindl, H 2004, 'Active Tool Support for Requirements Engineering Through RETH', 12th IEEE International Requirements Engineering Conference, Kyoto, Japan, September 6-10.

Kaindl, H, Kramer, S & Hailing, M 2001, 'An Interactive Guide Through a Defined Modelling Process', People and Computers XV, HCI 2001 and IHM 2001, Lille, France, September 10-14.

Kaiya, H, Saeki, M & Ochimizu, K 1995, 'Design of a Hyper Media Tool to support Requirements Elicitation Meetings', International Workshop on Computer-Aided Software Engineering, Toronto, Canada, July 10-14.

Karlsson, L, Dahlstedt, ÅG, Natt och Dag, J, Regnell, B & Persson, A 2002, 'Challenges in Market-Driven Requirements Engineering - an Industrial Interview Study', Eighth International Workshop on Requirements Engineering: Foundation for Software Quality, Essen, Germany, September 9-10.

Kasirun, ZM 2005, 'A Survey on the Requirements Elicitation Practices Among Courseware Developers', *Malaysian Journal of Computer Science*, vol. 18, no. 1, pp. 70-7.

Kassel, NW & Malloy, BA 2003, 'An Approach to Automate Requirements Elicitation and Specification', 7th IASTED International Conference on Software Engineering and Applications, Marina Del Rey, USA, November 3-5.

Kasser, JE 2004, 'The First Requirements Elucidator Demonstration (FRED) Tool', *Systems Engineering*, vol. 7, no. 3, pp. 243-56.

Kato, J, Komiya, S, Saeki, M, Ohnishi, A, Nagata, M, Yamamoto, S & Horai, H 2001, 'A Model for Navigating Interview Processes in Requirements Elicitation', Eighth Asia-Pacific Software Engineering Conference, Macao, China, December 4-12.

Kaufman, LD, Thebaut, S & Interrante, MF 1989, *System Modeling for Scenario-Based Requirements Engineering*, SERC-TR-33-F, SERC.

Kelly, G 1955, *The Psychology of Personal Constructs*, Norton, New York, USA.

Kendall, KE & Kendall, JE 2002, *Systems Analysis and Design*, Fifth edn, Prentice-Hall, USA.

Kirakowski, J & Corbett, M 1993, 'SUMI: the Software Usability Measurement Inventory', *British Journal of Educational Technology*, vol. 24, no. 3, pp. 210-2.

Kitchenham, B, Al-Khilidar, H, Babar, MA, Berry, M, Cox, K, Keung, J, Kurniawati, F, Staples, M, Zhang, H & Zhu, L 2006, 'Research methodology: Evaluating guidelines for empirical software engineering studies', ACM/IEEE International Symposium on Empirical Software Engineering, Rio de Janeiro, Brazil, September 21-22.

Kitchenham, B, Linkman, S & Law, D 1997, 'DESMET: a methodology for evaluating software engineering methods and tools', *Computing & Control Engineering Journal*, vol. 8, no. 3, pp. 120-6.

Kitchenham, B, Pickard, L & Pfleeger, SL 1995, 'Case Studies for Method and Tool Evaluation', *IEEE Software*, vol. 12, no. 4, pp. 52-62.

Kitchenham, BA 1996a, 'Evaluating Software Engineering Methods and Tool Part 1: The Evaluation Context and Evaluation Methods', *ACM SIGSOFT Software Engineering Notes*, vol. 21, no. 1, pp. 11-5.

---- 1996b, 'Evaluating Software Engineering Methods and Tool, Part 2: Selecting an appropriate evaluation method - technical criteria', *ACM SIGSOFT Software Engineering Notes*, vol. 21, no. 2, pp. 11-5.

Kitchenham, BA, Pfleeger, SL, Pickard, LM, Jones, PW, Hoaglin, DC, Emam, KE & Rosenberg, J 2002, 'Preliminary Guidelines for Empirical Research in

Software Engineering', *IEEE Transactions on Software Engineering*, vol. 28, no. 8, pp. 721-34.

Kitchenham, BA & Pickard, LM 1998a, 'Evaluating Software Eng. Methods and Tools, Part 10: Designing and Running a Quantitative Case Study', *ACM SIGSOFT Software Engineering Notes*, vol. 23, no. 3, pp. 20-2.

---- 1998b, 'Evaluating Software Engineering Methods and Tools, Part 9: Quantitative Case Study Methodology', *ACM SIGSOFT Software Engineering Notes*, vol. 23, no. 1, pp. 24-6.

Ko, Y, Park, S & Seo, J 2000, 'Web-based Requirements Elicitation Supporting System using Requirements Categorization', International Conference on Software Engineering and Knowledge Engineering, Chicago, USA, July 6-8.

Kotonya, G & Sommerville, I 1996, 'Requirements Engineering with Viewpoints', *BCS/IEE Software Engineering Journal*, vol. 11, no. 1, pp. 5-18.

---- 1998, *Requirements Engineering: Processes and Techniques*, John Wiley & Sons, Great Britain.

Krippendorf, K 2004, *Content Analysis: An Introduction to Its Methodology*, 2nd edn, Sage, Thousand Oaks, USA.

Kruchten, P 2003, *The Rational Unified Process: An Introduction*, 2nd edn, Addison-Wesley Professional, USA.

Krueger, RA 1994, *Focus Groups: A practical guide for applied research*, Sage, Thousand Oaks, USA.

Kurtz, T 2001, 'Ask Pete, Software Planning and Estimation through Project Characterization', International Symposium on Requirements Engineering, Toronto, Canada, August 27-31.

Kvale, S 1996, *Interviews: An Introduction to Qualitative Research Interviewing*, Sage, Thousand Oaks, USA.

LAAS 2006, *Laboratory for the Analysis and Architecture of Systems*, <http://www.laas.fr>.

Landsberger, HA 1958, *Hawthorne Revisited*, Cornell University.

Lauesen, S 2002, *Software Requirements: Styles and Techniques*, Addison-Wesley, Great Britain.

Lecoeuche, R, Mellish, C & Robertson, D 1998, 'A Framework for Requirements Elicitation through Mixed-Initiative Dialogue', Third International Conference on Requirements Engineering, Colorado Springs, USA, April 6-10.

Lee, JSK 1992, 'Quantitative versus Qualitative Research Methods - Two Approaches to Organization Studies', *Asia Pacific Journal of Management*, vol. 9, no. 1, pp. 87-94.

Leffingwell, D & Widrig, D 2000, *Managing Software Requirements: A Unified Approach*, Addison-Wesley, USA.

Leite, J & Gilvaz, A 1996, 'Requirements Elicitation Driven by Interviews: The Use of Viewpoints', 8th International Workshop on Software Specification and Design, Paderborn, Germany, March 22-23.

Liou, YI & Chen, M 1993, 'Integrating Group Support Systems, Joint Application Development, and Computer-Aided Software Engineering for Requirements Specification', 26th Annual Hawaii International Conference on System Sciences, Wailea, Hawaii, January 5-8.

Lloyd, WJ, Rosson, MB & Arthur, JD 2002, 'Effectiveness of Elicitation Techniques in Distributed Requirements Engineering', IEEE Joint International Conference on Requirements Engineering, Essen, Germany, September 9-13.

Long, RG, White, MC, Friedman, WH & Brazeal, DV 2000, 'The 'Qualitative' Versus 'Quantitative' Research Debate: A Question of Metaphorical Assumptions?' *International Journal of Value-Based Management*, vol. 13, no. 2, pp. 189-97.

Loucopoulos, P & Champion, REM 1990, 'Concept acquisition and analysis for requirements specification', *Software Engineering Journal*, vol. 5, no. 2, pp. 116-24.

Loucopoulos, P & Karakostas, V 1995, *Systems Requirements Engineering*, McGraw-Hill, London, UK.

Macaulay, L 1996, 'Requirements for Requirements Engineering Techniques', International Conference on Requirements Engineering, Colorado Springs, USA, April 15-18.

Macaulay, LA 1993, 'Requirements as a Cooperative Activity', IEEE Symposium on Requirements Engineering, San Diego, USA, January 4-6.

---- 1999, 'Seven-Layer Model of the Role of the Facilitator in Requirements Engineering', *Requirements Engineering*, vol. 4, pp. 38-59.

Maiden, N, Gizikis, A & Robertson, S 2004, 'Provoking Creativity: Imagine What Your Requirements Could be Like', *IEEE Software*, vol. 21, no. 5, pp. 68-75.

Maiden, N, Manning, S, Robertson, S & Greenwood, J 2004, 'Integrating Creativity Workshops into Structured Requirements Processes', Designing Interactive Systems, Cambridge, USA, August 1-4.

Maiden, NAM & Rugg, G 1996, 'ACRE: selecting methods for requirements acquisition', *Software Engineering Journal*, vol. 11, no. 3, pp. 183-92.

Maiden, NAM & Sutcliffe, AG 1993, 'Requirements Engineering by Example: an Empirical Study', IEEE International Symposium on Requirements Engineering, San Diego, USA, January 4-6.

Mannio, M & Nikula, U 2001, 'Requirements Elicitation Using a Combination of Prototypes and Scenarios', Workshop on Requirements Engineering, Buenos Aires, Argentina, November 22-23.

Martin, J 1991, *Rapid Application Development*, Prentice-Hall, New York, USA.

Martin, RC 2003, *Agile Software Development: Principles, Patterns, and Practices*, Prentice Hall, Upper Saddle River, USA.

McGraw, KL & Harbison-Briggs, K 1989, *Knowledge Acquisition: Principles and Guidelines*, Prentice-Hall, New Jersey, USA.

Meagher, P 2004, *Implement Bayesian inference using PHP, Part 1: Build intelligent Web applications through conditional probability*, 2005, <http://www-106.ibm.com/developerworks/web/library/wa-bayes1/>.

Mich, L, Anesi, C & Berry, DM 2005, 'Applying a Pragmatics-Based Creativity-Fostering Technique to Requirements Elicitation', *Requirements Engineering*, vol. 10, no. 4, pp. 262-75.

Microsoft 2005, *Visio*, <http://office.microsoft.com/en-us/FX010857981033.aspx>.

Mills, GE 2002, *Action Research: A Guide for the Teacher Researcher*, 2nd edn, Prentice Hall, Upper Saddle River, USA.

Moore, MJ & Shipman, FM 2000, 'A Comparison of Questionnaire-Based and GUI-Based Requirements Gathering', IEEE International Conference on Automated Software Engineering, Grenoble, France, September 11-15.

Mullery, GP 1979, 'CORE - A Method for Controlled Requirement Specification', International Conference on Software Engineering, Munich, Germany, September.

Mumford, E 1995, *Effective Systems Design and Requirements Analysis - The ETHICS Approach*, MacMillan Press, Basingstoke, UK.

Myers, MD 2007, *Qualitative Research in Information Systems*, 2007, <http://www.qual.auckland.ac.nz>.

Neill, CJ & Laplante, PA 2003, 'Requirements Engineering: The State of the Practice', *IEEE Software*, vol. 20, no. 6, pp. 40 - 5.

Nielsen, J, Clemmensen, T & Yssing, C 2002, 'Getting access to what goes on in people's heads?: reflections on the think-aloud technique', Second Nordic

Conference on Human-Computer Interaction, Aarhus, Denmark, October 19-23.

Nunamaker, JF, Briggs, RO & Mittleman, DD 1996, 'Lessons from a Decade of Group Support Systems Research', 29th Annual Hawaii International Conference on System Sciences, Maui, Hawaii, January 3-6.

Nuseibeh, B & Easterbrook, S 2000, 'Requirements Engineering: A Roadmap', The Future of Software Engineering, Limerick, Ireland, June 4-11.

Nuseibeh, B, Finkelstein, A & Kramer, J 1996, 'Method Engineering for Multi-Perspective Software Development', *Information and Software Technology Journal*, vol. 38, no. 4, pp. 267-74.

Nuseibeh, B, Kramer, J & Finkelstein, A 1994, 'A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification', *Transactions on Software Engineering*, vol. 20, no. 10, pp. 760-73.

Olson, H 2006, *Quantitative "versus" Qualitative Research: The Wrong Question*, <http://www.ualberta.ca/dept/slis/cais/olson.htm>.

OMG 2004, *UML 2.0 Superstructure Specification*.

OPEN Process Framework 2007, *OPEN Process Framework*, 2007, <http://www.opfro.org>.

Oppenheim, AN 1992, *Questionnaire design, interviewing and attitude measurement*, Pinter Publishers, London, UK.

Orlikowski, WJ & Baroudi, JJ 1991, 'Studying Information Technology in Organizations: Research Approaches and Assumptions', *Information Systems Research*, vol. 2, pp. 1-28.

Osborn, AF 1979, *Applied Imagination*, Charles Scribner's Sons, New York, USA.

Page-Bucci, H 2003, *The value of Likert scales in measuring attitudes of online learners*, <http://www.hkadesigns.co.uk/websites/msc/reme/likert.htm>.

Palmer, JD & Fields, NA 1992, 'An Integrated Environment for Requirements Engineering', *IEEE Software*, vol. 9, no. 3, pp. 80-5.

Parnes, SJ 1967, *Creative behavior guidebook*, Charles Scribner's Sons, New York, USA.

Patton, MQ 2001, *Qualitative Research & Evaluation Methods*, 3rd edn, Sage Publications, USA.

Peffers, K, Gengler, CE & Tuunanen, T 2003, 'Extending critical success factors methodology to facilitate broadly participative information systems planning', *Journal of Management Information Systems*, vol. 20, no. 1, pp. 51-85.

Pfleeger, SL 1991, *Software Engineering: The Production of Quality Software*, Second edn, Macmillan, Sydney, Australia.

Playle, G & Schroeder, C 1996, 'Software Requirements Elicitation: Problems, Tools, and Techniques', *CROSSTALK The Journal of Defense Software Engineering*, vol. 9, no. 12, pp. 19-24.

Pohl, K 1996, *Process Centered Requirements Engineering*, Wiley, New York, USA.

Pohl, K, Assenova, P, Doemges, R, Johannesson, P, Maiden, N, Plihon, V, Schmitt, J-R & Spanoudakis, G 1994, 'Applying AI Techniques to Requirements Engineering: The NATURE Prototype', International Conference on Software Engineering, Edinburgh, UK, May 23-28.

Potts, C, Takahashi, K & Anton, AI 1994, 'Inquiry-Based Requirements Analysis', *IEEE Software*, vol. 11, no. 2, pp. 21-32.

QSR International 2006, *NVivo*, <http://www.qsrinternational.com>.

QuestionPro 2005, *QuestionPro*, <http://www.questionpro.com>.

Raghavan, S, Zelesnik, G & Ford, G 1994, *Lecture Notes on Requirements Elicitation*, CMU/SEI-94-EM-10, Carnegie Mellon University, Software Engineering Institute.

Ralyté, J, Deneckère, R & Rolland, C 2003, 'Towards a Generic Model for Situational Method Engineering', International Conference on Computer-Aided Software Engineering, Klagenfurt, Austria, June 16-20.

Ralyté, J & Rolland, C 2001, 'An Assembly Process Model for Method Engineering', International Conference on Computer-Aided Software Engineering, Interlaken, Switzerland, June 4-8.

Rayson, P, Garside, R & Sawyer, P 2000, *Assisting Requirements Recovery from Legacy Documents*, Computing Department, Lancaster University.

Rea, LM & Parker, RA 2005, *Designing and Conducting Survey Research: A Comprehensive Guide*, 3rd edn, Jossey-Bass.

REDEST 2003, *Requirements Engineering Best Practice Case Book*.

Research Methods Knowledge Base 2006, *Likert Scaling*, <http://www.socialresearchmethods.net/kb/scallik.htm>.

Reubenstein, HB & Waters, RC 1991, 'The Requirements Apprentice: Automated Assistance for Requirements Acquisition', *IEEE Transactions on Software Engineering*, vol. 17, no. 3, pp. 226-40.

Richardson, J, Ormerod, TC & Shepherd, A 1998, 'The Role of Task Analysis in Capturing Requirements for Interface Design', *Interacting with Computers*, vol. 9, no. 4, pp. 367-84.

RMTrak 2005, *RMTrak*, <http://www.rmtrak.com/>.

Robertson, J 2002, 'Eureka! Why Analysts Should Invent Requirements', *IEEE Software*, vol. 19, no. 4, pp. 20-2.

Robertson, S & Robertson, J 1999, *Mastering the Requirements Process*, Addison-Wesley, Great Britain.

Rolland, C & Prakash, N 2000, 'From conceptual modelling to requirements engineering', *Annals of Software Engineering*, vol. 10, p. 151–76.

Rolland, C, Souveyet, C & Ben Achour, C 1998, 'Guiding Goal Modeling Using Scenarios', *IEEE Transactions on Software Engineering*, vol. 24, no. 12, pp. 1055-71.

Rolston, D 2005, *LAMP, MySQL/PHP Database Driven Websites - Parts I, II, and III*, <http://www.phpfreaks.com/tutorials/>.

Rook, P 1986, 'Controlling software projects', *Software Engineering Journal*, vol. 1, no. 1, pp. 7-16.

Roseman, M 2005, *TeamWave Retrospective*, <http://www.markroseman.com/teamwave/>.

Royce, W 1970, 'Managing the Development of Large Software Systems', IEEE WESCON, Los Angeles, August 25-28.

Sadler, C & Kitchenham, BA 1996, 'Evaluating Software Engineering Methods and Tool, Part 4: The influence of human factors', *ACM SIGSOFT Software Engineering Notes*, vol. 21, no. 5, pp. 11-3.

Saeki, M 1995, 'Communication, Collaboration and Cooperation in Software Development - How Should We Support Group Work in Software Development?' Asia Pacific Software Engineering Conference, Brisbane, Australia, December 6-9.

---- 2003, 'CAME : The First Step to Automated Method Engineering', Workshop on Process Engineering for Object-Oriented and Component-Based Development, Anaheim, USA, October 26-30.

Saeki, M, Tsuchida, M & Nishiue, K 2000, 'Supporting tool for assembling software specification and design methods', 24th Annual International Computer Software and Applications Conference, Taipei, Taiwan, October 25-27.

Satterfield, B 2006, *Visual Presentations Made Easy With Diagramming Software*, 2006, <http://www.techsoup.org>.

Satzinger, JW, Jackson, RB & Burd, SD 2002, *Systems Analysis and Design in a Changing World*, Second edn, Course Technology/Thomson Learning, Canada.

Sauro, J & Kindlund, E 2005, 'A Method to Standardize Usability Metrics Into a Single Score', CHI 2005, Portland, USA, April 2-7.

Scenario Plus 2007, *Scenario Plus*, 2007, <http://www.scenarioplus.org.uk>.

Schalken, J, Brinkkemper, S & van Vlieet, H 2004, 'Assessing the Effects of Facilitated Workshops in Requirements Engineering', 8th Conference on Evaluation & Assessment in Software Engineering, Stevenage, UK, May 23-24.

Scott, W & Cook, SC 2003, 'An Architecture for an Intelligent Requirements Elicitation and Assessment Assistant', 13th Annual International Symposium - INCOSE 2003, Crystal City, USA, July 1-3.

Serena 2005, *RTM (Requirements Traceability Management)*, <http://www.serena.com/Products/rtm/home.asp>.

Sharp, H, Finkelstein, A & Galal, G 1999, 'Stakeholder identification in the requirements engineering process', Tenth International Workshop on Database and Expert Systems Applications, Florence, Italy, September 1-3.

Shaw, MLG & Gaines, BR 1995, 'Comparing Constructions through the Web', Computer Support for Collaborative Learning, September 1.

---- 1996, 'Requirements acquisition', *Software Engineering Journal*, vol. 11, no. 3, pp. 149-65.

Shelly, GB, Cashman, TJ & Rosenblatt, HJ 2003, *Systems Analysis and Design*, Fifth edn, Course Technology/Thomson, USA.

Shimakage, M & Hazeyama, A 2004, 'A Requirement Elicitation Method in Collaborative Software Development Community', 5th International Conference on Product Focused Software Process Improvement, Kyoto-Nara, Japan, April 5-8.

Sofea 2005, *Profesy*, <http://www.sofeainc.com/product_overview.htm>.

Sommerville, I 2001, *Software Engineering*, 6th edn, Addison-Wesley, USA.

Sommerville, I & Sawyer, P 1997, *Requirements Engineering: A Good Practice Guide*, John Wiley & Sons, Great Britain.

Sommerville, I, Sawyer, P & Viller, S 1998, 'Viewpoints For Requirements Elicitation: A Practical Approach', IEEE International Conference on Requirements Engineering, Colorado Springs, USA, April 6-10.

SPSS Inc. 2006, *SPSS*, <http://www.spss.com>.

Stake, R 1995, *The art of case research*, Sage, Thousand Oaks, USA.

Stapleton, J 2002, *Framework for Business Centred Development:*, DSDM Consortium, UK.

Stemler, S 2001, 'An overview of content analysis', *Practical Assessment, Research & Evaluation*, vol. 7, no. 17.

Straub, D, Gefen, D & Boudreau, M-C 2005, *The ISWorld Quantitative, Positivist Research Methods Website*, 2005, <http://dstraub.cis.gsu.edu:88/quant/>.

Subramaniam, UV 1999, 'An Event, Activity and Process Based Methodology for Requirements Elicitation and its Application to an Educational Information System', Sixth Asia Pacific Software Engineering Conference, Takamatsu, Japan, December 7-10.

Sutcliffe, A 1997, 'A Technique Combination Approach to Requirements Engineering', IEEE International Symposium on Requirements Engineering, Annapolis, USA, January 6-10.

Sutcliffe, A & Maiden, N 1998, 'The Domain Theory for Requirements Engineering', *IEEE Transactions on Software Engineering*, vol. 24, no. 3, pp. 174-96.

Sutcliffe, A & Ryan, M 1998, 'Experience with SCRAM: A SCenario Requirements Analysis Method', IEEE Third International Conference on Requirements Engineering, Colorado Springs, USA, April 6-10.

Sutcliffe, AG, Maiden, NAM, Minocha, S & Manuel , D 1998, 'Supporting Scenario-based Requirements Engineering', *IEEE Transactions on Software Engineering*, vol. 24, no. 12, p. 1072–88.

Taylor, MA & Urban, JE 1994, 'A method for evaluating software engineering environments', Eighteenth Annual International Computer Software and Applications Conference, Los Alamitos, USA, November 9-11.

TeamWave 2005, *TeamWave*, <http://www.teamwave.com>.

Telelogic 2005, *DOORS*, <http://www.telelogic.com/corp/products/doors/index.cfm>.

Thayer, RH & Dorfman, M 1987, *Software Requirements Engineering*, 2nd edn, IEEE Computer Society Press, Los Alamitos, USA.

The Standish Group 1994, *The CHAOS Report*.

---- 2003, *CHAOS Chronicles v3.0*.

Tigris 2005, *ArgoUML*, <http://argouml.tigris.org/>.

Togneri, DF, de Almeida Falbo, R & de Menezes, CS 2002, 'Supporting Cooperative Requirements Engineering with an Automated Tool', Workshop on Requirements Engineering, Valencia, Spain, November 11-12.

Tropos 2006, *Tropos - Requirements-Driven Development for Agent Software*, <http://troposproject.org/>.

Truereq 2005, *TRUEreq*, <http://www.truereq.com/>.

Tuunanen, T 2003, 'A New Perspective on Requirements Elicitation Methods', *Journal of Information Technology Theory and Application*, vol. 5, no. 3, pp. 45-62.

Tuunanen, T & Rossi, M 2004, 'Engineering a Method for Wide Audience Requirements Elicitation and Integrating It to Software Development', 37th Annual Hawaii International Conference on System Sciences, Big Island, Hawaii, January 5-8.

University of Toronto 2006, *Organization Modelling Environment (OMG)*, <http://www.cs.toronto.edu/km/ome/>.

van Lamsweerde, A 2000, 'Requirements Engineering in the Year 00: A Research Perspective', International Conference on Software Engineering, Limerick, Ireland, June 4-11.

Venable, JR & Travis, J 1999, 'Using a Group Support System for the Distributed Application of Soft Systems Methodology', 10th Australasian Conference on Information Systems, Wellington, New Zealand, December 1-3.

Wang, X & Loucopoulos, P 1995, 'The Development of Phedias: a CASE Shell', Seventh International Workshop on Computer-Aided Software Engineering, Toronto, Canada, July 10-14.

Wiegers, KE 2003, *Software Requirements*, Second edn, Microsoft Press, USA.

---- 2006, *More About Software Requirements*, Microsoft Press, USA.

---- 2007, *Process Impact*, 2007, <http://www.processimpact.com>.

Windle, DR & Abreo, LR 2002, *Software Requirements Using the Unified Process*, Prentice Hall PTR, Upper Saddle River.

Wixon, D & Ramey, J (eds) 1996, *Field Methods Casebook for Software Design*, John Wiley & Sons, USA.

Wood, DP, Christel, MG & Stevens, SM 1994, 'A Multimedia Approach to Requirements Capture and Modeling', First International Conference on Requirements Engineering, Colorado Springs, USA, April 18-22.

Wood, J & Silver, D 1995, *Joint Application Development*, John Wiley & Sons, New York, USA.

Wycoff, J 1991, *Mindmapping: Your Personal Guide to Exploring Creativity and Problem-Solving*, Berkley Books, New York, USA.

Yeaple, RN 1992, 'Why are small R&D organizations more productive?' *IEEE Transactions on Engineering Management*, vol. 39, no. 4, pp. 332-46.

Yeates, D & Wakefield, T 2004, *Systems Analysis and Design*, Second edn, Financial Times/Prentice Hall, Sydney, Australia.

Yin, RK 1994, *Case Study Research: Design and Methods*, Second edn, Sage, Thousand Oaks, USA.

Young, E 2004, *Artificial Neural Network in PHP*, 2005, <http://coding.mu/archives/2004/03/19/artificial_neural_network_in_php/>.

Yourdon, E 1989, *Modern Structured Analysis*, Prentice Hall, Englewood Cliffs, USA.

Yu, ESK 1997, 'Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering', Third IEEE International Symposium on Requirements Engineering, Washington D.C., USA, January 5-8.

Zave, P 1997, 'Classification of Research Efforts in Requirements Engineering', *ACM Computing Surveys (CSUR)*, vol. 29, no. 4, pp. 315-21.

Zave, P & Jackson, M 1997, 'Four dark corners of requirements engineering', *ACM Transactions on Software Engineering and Methodology*, vol. 6, no. 1, pp. 1-30.

Zeroual, K 1991, 'A Knowledge-based Requirements Acquisition System', 6th Annual Knowledge-Based Software Engineering Conference, Syracuse, USA, September 22-25.

Zowghi, D 1999, 'A Logic-Based Framework for the Management of Changing Software Requirements', Doctoral Dissertation thesis, Macquarie University.

Zowghi, D & Coulin, C 2005, 'Requirements Elicitation: A Survey of Techniques, Approaches, and Tools', in A Aurum & C Wohlin (eds), *Engineering and Managing Software Requirements*, Springer, USA.

Zowghi, D & Paryani, S 2003, 'Teaching requirements engineering through role playing: lessons learnt', 11th IEEE International Requirements Engineering Conference, Monterey Bay, USA, September 8-12.