ISSN 1744-1986

Technical Report N° 2006/ 26

A comparison of Problem Frames, a problem-based method, and KAOS, a goal-based method, for Requirements Analysis within a financial environment

Michele Doran

30 September, 2006

Department of Computing
Faculty of Mathematics, Computing and Technology
The Open University

Walton Hall, Milton Keynes, MK7 6AA
United Kingdom

http://computing.open.ac.uk

A comparison of Problem Frames, a problem-based method, and KAOS, a goal-based method, for Requirements Analysis within a financial environment

A dissertation submitted in partial fulfilment of the requirements
for the Open University's
Master of Science Degree in Computing for Commerce and Industry

Michele Doran[redacted]
8 March 2007
Word Count: 16,751

Preface

As a software developer, there is a definite sense of satisfaction when software is delivered to the users without a returned list of errors and requirements that were overlooked. The task of Requirements Analysis is extremely important and I hope that this work contributes to the number of successfully delivered projects.

Throughout this project I have received much encouragement from family, friends and colleagues and I would like to thank them all, as this has been very important to me. However, I would especially like to thank the following people for the large part that they have played in the completion of this project:

my tutor, Joan Jackson, for her support, advice and extremely useful feedback;

my designer/analyst and system tester colleagues for their lively and thoughtful review meetings;

my sister Jules, who has also doubled as my librarian;

my fiancé Mike, for his constant support, encouragement and conscientious proof reading that has resulted in a more readable dissertation.

Thank you all very much!

Table of Contents

List of Tables

List of Figures

Abstract

This research reviews the two Requirement Analysis methods of Problem Frames and KAOS and then applies them to a case study within a financial environment. Both methods focus on the problem domain and have techniques to ensure that the task of Requirements Analysis is carried out in a logical manner that should help to produce complete, consistent and unambiguous requirements. The intention of the research is to make recommendations and produce guidelines for use within financial environments to help to improve the quality of requirements specifications.

Specific characteristics of financial environments are reviewed to determine what makes them require special attention in comparison to other, more mechanical environments. As well as general characteristics applicable to all environments, specific characteristics were found that do require special focus. These are complexity of functionality, adaptability to changing requirements to comply with laws and regulations, keeping a competitive edge by delivering quickly to meet the demands of the market and finally, the existence of legacy systems.

Existing literature covering the two methods is reviewed to analyse previous descriptions of their application and the advantages and disadvantages experienced. The application to a case study then allowed the comparison of these findings with first hand experience in a financial environment. The similarities were many showing that the advantages of a particular method did carry into the financial environment, although many disadvantages did as well. The requirements specifications produced during the two methods were reviewed by experienced designers /analysts and system testers and their feedback incorporated into the conclusions of the study.

Using the full set of conclusions, guidelines and recommendations have been produced, reviewed and then refined. The key findings, that aim to address the financial environment

specific characteristics, are that at specific stages the models and analysis of the requirements should be reviewed with the key stakeholders to try to address the issue of complexity and that the financial environment should keep a catalogued and classified store of past experience to encourage reuse in the early stages of the development lifecycle to aid the delivery time of required software. Problem Frames and KAOS both support these processes but it may be a combination of the two methods which provides the best approach to Requirements Analysis in a financial environment.

Although the research is based on a single-case study, the results and conclusions have been made for all financial environments by drawing on their general characteristics. The resulting guidelines are hoped to be of use to improve Requirements Analysis within financial environments.

Chapter 1 Introduction

This chapter contains an overview of the problems associated with the task of Requirements Analysis. It then discusses the use of guidelines and an introduction to the background of this project. Finally, the project aims and structure are presented.

1.1 Overview of Problems Associated with Requirements Analysis

Requirements Analysis 'is one of the main life-cycle activities responsible for making software development a disciplined engineering process.' (The Open University, 1997, p6). With this importance placed on the task of Requirements Analysis, it is not surprising that many academic researchers have studied this area with the view of formalising and structuring the whole process. There are many documented approaches, methods and techniques for Requirements Analysis and Specification, the Structured and Object-Oriented approaches being the most commonly known. To name a few more, in no particular order, Data Structured Systems Development (DSSD) / Warnier-Orr Methodology, Formal Methods like Z, the Inquiry-Based approach, Problem Frames, the Reference Model approach, Goal-Based approaches, the Volere template, Affirm and Gist.

However, despite the variety of approaches, methods and techniques that are available, an early observation by Bell and Thayer (1976) is still being observed in commercial software development today, which is that inadequate, inconsistent, incomplete or ambiguous requirements are numerous and have a critical impact on the quality of the software developed. Chen et al. (2002, p221) highlight the importance of 'complete, correct, consistent and unambiguous' requirement specification documents as 'any defect will be propagated to subsequent phases of the [development life cycle]'. Robertson and Robertson (1999, p27) argue that the 'cost of good requirements gathering and systems analysis is minor compared

to the cost of poor requirements.' They go on to state that over half of software errors originate in requirements gathering and analysis tasks. This is partly attributed to software developers beginning to build a solution prior to the full problem being known. However, it also originates from incorrect or incomplete requirements being gathered. Pressman (1994) argues that one of the problems of software development is that customer dissatisfaction with the delivered system is encountered more often than is acceptable. He attributes this to development projects frequently being undertaken with only a vague indication of customer requirements. He also comments that communication between customers and software developers is often inadequate. More recently, Jackson (2001) argues that software development methods fail because they are advertised as being the best approach to tackle all development problems, instead of stating which category of problem or development area they are specifically suited to. As an example of the size of this problem, Ashry and Taylor, (2000, p2) state that one study 'reported that after project completion, 76 information systems developed in 26 organisations all required some form of revision to even partially fulfil management's information needs'.

1.2 Choosing a Requirements Analysis Approach

Guidelines have been presented to help software developers and organisations to choose their Requirements Analysis approach. However, there is not an overall guideline for this task as different software environments appear to require different approaches. Davis (1993) provides an example of guidelines, however he also highlights that no single approach might be adequate and a combination of approaches might prove more complete especially for complicated software. The idea of combinations of approaches for complex systems is also argued by Jackson (1995). Further, Wing (1988, p76) clearly puts the onus on the authors of requirement specification methods to explain their applicability and best practice use, stating

‘It is the responsibility of an advocate of a particular specification method to tell potential users not only what the method is good for but also what it is not good for. Students (and readers) should not expect the method to be suitable for classes of applications and properties outside of the method’s intended ones. However, students of a particular specification method should also not be expected to guess what those suitable classes are’.

There appears to be a level of agreement that no sole approach, method or technique has been documented yet that will be equally applicable and useful to all environments and problems for requirements specification. It is therefore very important that specific, relevant guidelines and procedures exist for developers and analysts to decide which approach, method or technique, or combination of methods and techniques, should be used in their environment.

1.3 Project Overview

To summarise the above, with such a choice and diversity of approaches coupled with the task’s importance, choosing the most applicable approach for a specific software development environment requires investigation. This project will be focusing on financial environments and their specific characteristics.

An extra consideration for this project is that when a problem or need is raised the software engineer is required to initially analyse the problem to be solved and not to look straight at possible solutions. On reviewing different approaches and techniques Jackson (1994, p251) argues that the ‘concentration on solutions is more widespread than may be recognised. Some methods claim to be analysing or structuring the problem when in fact they place all their emphasis on describing a solution.’ He goes on to compare the approach of engineers from established engineering branches and urges that to be in line with these professional engineers, software developers must focus on the problem domain.

Two such methods which do focus on the problem domain are goal-based and problem-based methods. Therefore an example of each of these two methods will be used for this project to investigate their use when carrying out Requirements Analysis within a financial environment. The chosen problem-based method is Problem Frames and the chosen goal-based method is KAOS (Knowledge Acquisition in AutOmated Specifications). Both were chosen as they are well documented examples of the two styles.

The task of Requirements Analysis will be broken down into the steps of requirements elicitation, modelling and analysis so that each of the two chosen methods can be reviewed and analysed across the range of steps involved in the task of Requirements Analysis. To avoid confusion between the umbrella task of Requirements Analysis and the specific task of analysis of the requirements, the former will always be written using initial capitals i.e. Requirements Analysis. Breaking the umbrella task into these three steps will provide a framework to aid the assessment and analysis of each method, highlighting advantages and disadvantages of each at particular stages. Within this project, these three steps will be defined as below.

- **Elicitation**

When referring to this step I will be describing the gathering of information about the problem area and requirements from various stakeholders. When successfully carried out, this step provides a full set of information about the problem domain.

- **Modelling**

When referring to this step I will be looking at the different ways of illustrating and describing the information gathered regarding the problem domain features, requirements and agents. It will be analytic modelling. When successfully carried out, this step provides documentation that is understood by all stakeholders and is a useful discussion point for

agreeing the requirements.

- **Analysis**

When referring to this step I will be reviewing how all of the information gathered can be analysed in order to determine that it is complete and whether the analysis tools can be utilised to highlight conflicts or ambiguities and be used to prioritise requirements. When successfully carried out, this step will highlight areas where further clarification or analysis is needed to ensure complete requirements are specified.

1.4 Project Aims

An aim of my project is to review and compare the two Requirements Analysis methods of Problem Frames and KAOS within a financial environment. This comparison of the two methods will focus on any advantages or disadvantages of each with specific emphasis on the steps of elicitation, modelling and analysis within a financial environment.

A further aim is to compare my results with those that are already documented in existing research on this topic, drawing conclusions on whether financial environments need special consideration from other environments when using either of these two methods. Another consideration will be whether my findings agree with any similar research that has already been documented.

Finally, I aim to find out whether any recommendations or guidelines can be made to advise when to use each method, or how parts of each method might be utilised to the best advantage in a financial environment. The guidelines and procedures will then be evaluated to determine how well they can be used by requirements engineers and designers working in a financial environment to create a positive impact on the quality of the requirements specification and ultimately the quality of the software developed. It is an aim that the output of this research

will be utilised within my financial environment, and in fact any financial environment, to enrich the Requirements Analysis stage of the software development life cycle and thus lead to better quality software that meets user requirements. As noted earlier, if half of the software errors arise in the requirements gathering stage, improving the methods used will have a positive impact on the final product. However, given the limitation that this research project is only based on a single-case study, any conclusions may also be limited.

The research question for this dissertation is therefore: which Requirements Analysis method, drawn from the existing common methods of Problem Frames and KAOS, looks to be more successful for the steps of requirements elicitation, modelling and analysis within a financial environment?

1.5 Structure of the Dissertation

Initially I will present an up to date review of current literature regarding the general Requirements Analysis characteristics that are applicable across environments; characteristics specifically for financial environments; experience of the application of the Problem Frame method; and the experience of the application of the KAOS method. The Problem Frame and KAOS methods have each already been used commercially and this usage has been analysed and documented, highlighting particular strengths and weaknesses in areas during requirements specification steps. Also, types of problem-based and goal-based methods have already been compared with each other in different ways and in a variety of environments.

I will then provide an overview of how to apply the two methods practically to a case study in a financial environment. I aim to contribute to the knowledge of this topic by comparing my conclusions with the conclusions already provided by current research. The aim is to identify whether financial environments need to be considered separately to other environments or

whether they can be considered alongside other environments. My research will highlight any special considerations that financial environments might require.

Finally, the resulting recommendations or guidelines for when to use each method, or how parts of each method might be utilised to the best advantage in a financial environment will be presented together with my conclusions and suggestions for further research in this area.

Chapter 2 Literature Review

This Chapter is divided into two distinct parts. In the first part I will initially highlight the general Requirements Analysis considerations that are independent of the development environment. Then I will highlight those characteristics specific to financial environments.

The second part of this Chapter will focus on the two Requirements Analysis methods chosen for review within a financial environment: the problem-based method, Problem Frames, and the goal-based method, KAOS. Recent research will be presented highlighting documented problem situations / application environments and the advantages and disadvantages found for each method in respect of the steps of elicitation, modelling and analysis. As previously highlighted, the specific characteristics of financial systems will be considered alongside general factors that are applicable across environments.

2.1 General Requirements Analysis Considerations

Independent of the environment that the software development is taking place in or the Requirements Analysis method being used, Pressman (1994) listed four fundamental principles that are still valid today.

- The problem domain must be understood.
- Models representing information, function and behaviour should be created.
- The problem domain should be divided into smaller, more manageable parts.
- Information should progress from high level to low level detail as the process evolves.

Further more, taking into consideration some ideas behind Davis' (1993) guidelines the following factors should also be considered:

- Utilise the current skills of the analysts involved.
- Consider all possible approaches before making a decision.
- Developers should gather experience and ability to choose the right technique at the right time.

In Ashry and Taylor (2000, p5) one of the conclusions drawn about their Requirements Analysis method (Grounded Theory) was that it was 'time-consuming and complex…mastering the method is difficult, and a mentor is often recommended.' They go on to argue that the time spent on mastering the method does balance against the quality gained and hence less post implementation changes are required. However, they did not present any metrics to prove this.

Based on these general considerations I will take forward the following points to help evaluate the two methods under study.

- Time must be spent gaining knowledge about the problem domain.
- Models should be created.
- The problem should be divided into manageable parts.
- Information should start off high level and increase in detail.
- The principles behind the method must be familiar or easy to learn so that the analyst knows how and when to use it.

2.2 Characteristics of Financial Environments

There is only a limited amount of literature available based on financial environments and so my literature review is focused in a large part on industry magazine articles. Also drawing on my own experience in the financial environment I have found that the main characteristics that need to be considered when carrying out Requirements Analysis, on top of the general environment considerations, are: complexity of functionality; the need to comply with audit and regulatory requirements; the need to keep a competitive edge by delivering the latest functionality quickly; and the existence of legacy systems.

Complexity - Financial systems are typically built to operate in complex problem domains where the environment knowledge required to fully understand the problem is very specialised and usually requires in depth accounting and book keeping knowledge. This means that the development analysts need to work very closely with the user stakeholders in order to obtain correct and complete requirements. A specific example highlighting the difficulty of Requirements Analysis of financial systems is the London stock exchange's TAURUS project of 1993 that was very publicly cancelled before being launched. An Economist article (Economist, 1993) when reviewing the process for TAURUS's replacement, the CREST system, highlighted the complexity of the requirements in terms of basic functionality and the decision over which international laws it would comply with and whether it would go too far and no longer comply with company law. This system was to have a large customer / user group and each had their specific requirements. Deciding which requirements to include and which to omit from the specification proved too big and complex a task.

Audit and Regulatory Requirements - There are internal and external audit requirements and international regulatory requirements which must be adhered to in order for financial institutions to gain permission to operate in a country. Recent examples are Sarbanes-Oxley Controls and the OFAC regulation. In the UK, the Financial Services Authority (FSA) issues

information regarding these regulations and is responsible for highlighting possible future regulations. In the near future there are emerging regulations like Serial Payments and the Markets in Financial Instruments Directive (Mifid) that first require financial interpretation and then need to be activated by the financial systems. As well as having strict deadlines, these requirements are usually very complex. Additionally, for large global financial companies, their systems often are required to be operational in many countries across the world, complying with many different local requirements and laws. A recent article in Computing magazine documented the findings of a senior security analyst at Forrester Research who found that only 2% of financial service firms are prepared for regulations that will be in place in 2007 (Computing, 2006a). As well as being complex and time dependent for delivery to market, this characteristic requires existing development projects to adapt to changing requirements as the problem domain is constantly moving and the system must always comply with new regulations and laws. This aspect will therefore be selected from this characteristic.

Competitive Edge – Financial companies in today's competitive world, just as many other businesses, have to comply with customer choice over product range in a short space of time. Dehong (2001) explains his 'pushing' model as the financial industry introducing financial products to customers and his 'pulling' model as customers who tell financial institutions what they need. He then states that it is the 'pulling force that urges financial corporations to innovate and re-engineer' (Dehong, 2001, p92). Financial companies therefore need to be able to react quickly to the changes in the technology market as well as the financial product market. This speed of change can mean that any lengthy development projects will again face constantly changing requirements and so techniques to be able to deliver new functionality quickly need to be in use. This leads to the technique of reuse, which allows development teams to make use of past project effort and so reduce the development time and deliver quickly.

Characteristic	Implications for Requirements Analysis
Complexity	Development analysts need to work very closely with the user stakeholders to obtain correct and complete requirements.
Regulations	Ability to adapt to changing requirements to comply with new regulations and laws.
Competitive edge	Techniques are required to be able to deliver new functionality quickly. The technique of reuse allows development teams to make use of past project effort.
Legacy systems	Reliance on older skill set and reliable documentation.

Table 2.1 Financial Environment Characteristics

Legacy Systems – In contrast to the competitive edge characteristic, there is often a strong reliance on legacy systems to perform specific functions in the processing chain. This can often be accredited to the fact that most development projects focus on customer facing systems to the detriment of the back-office systems used by the financial institution's staff. Another factor is that most financial institutions have to interact with specific countries central banks and regulatory offices who in themselves are not subject to the market forces that often drive the development of new systems. This means that legacy systems are left in place for many years. A recent example of the issues that legacy systems cause was made very public in the year 2000, when the demand for developers with 'older' development language skills was high. Working alongside legacy systems means that developer skills have to be maintained in more traditional technologies. There is also a reliance on good documentation that is very descriptive of the legacy system, its capabilities and its technologies. Barclays Bank is a recent example of a financial institution that is

implementing projects to replace legacy systems in an attempt to modernise their infrastructure (Computing, 2006b). As well as being driven by support and maintenance costs, the upgraded systems will allow a greater variety of products to be sold to customers.

The four main characteristics of financial environments have been summarised in Table 2.1.

2.3 Experiences Using Problem Frames

The concept of Problem Frames was first published by Jackson as early as 1994, (Jackson, 1994) and his essay edited by Roscoe (Jackson, 1994a). A concise definition of the concept of Problem Frames was provided by Cox et al. (2004, p754) stating that 'A *problem frame* is a definition of a recurring class of problem in software development, akin to a design pattern but for describing problems, not solutions'. It is a way of classifying problems by identifying the domain and characteristics and then fitting these to an existing template, although the fit must be exact in order for the specific problem frame to be applicable and useful in finding a solution.

Jackson attributes his idea of Problem Frames to an earlier idea, that a problem can be analysed into its principle parts (Polya, 1957). Polya identified problem classes, one such being 'problems to find or construct'.

As an example, consider the following problem.

Given the radius of a circle is length a, find the area of that circle. The principle parts of this problem are:

the *data* (length a)

the *unknown* (the area of the circle) and

the *condition* (that the circle radius be of the given length).

The *solution task* is to calculate the *unknown* so that it bears the relationship to the *data* described by the *condition*.

Jackson (1995a, p4) argues that problems in the software development world are very similar to Polya's problems to find or construct. He breaks the problems down into principle parts as follows:

the *application domain*;

the *machine*; and

the *requirement*.

The *solution task* is to construct the *machine* so that it ensures that the *requirement* holds in the *application domain*.

The concept of Problem Frames has been extensively developed (Jackson, 1995; Jackson, 2001) and as Jackson developed his method he defined a set of problem frames. Each problem frame classification is distinct and can be distinguished from the others by its requirements, domain characteristics, involvement in the problem and its 'problem concern'. The five basic problem frames are set out below as described by Jackson (2001).

- **Commanded behaviour** – this problem is to build a machine that will accept an operator's commands and impose control on some part of the physical world to satisfy those commands.

- **Information display** – this problem is to build a machine that will obtain desired information from the real world and present it at the required place and in the required form. This frame interacts with the real world but should not be used if the

displayed information has to be acted upon by the machine as this would then call for the required behaviour frame.

- **Required behaviour** – this problem is to build a machine that will control the behaviour of some part of the physical world so that set conditions are satisfied. This is the richest frame as it interacts with the real world the most, considering behavioural aspects.

- **Simple workpieces** – this problem is to build a machine that allows a user to create and edit a set of computer-process-able text or graphic objects so that they can be used for printing, copying or other ways as the user requires. This frame should be chosen when the input commands of the user are to be considered and the mapping between the input and output is altered as a consequence. It should not be used if the reasons behind why the user behaves in this manner is of consideration.

- **Transformation** – this problem is to build a machine that will accept computer – readable input files whose data must be transformed to give required output files. This is the shallowest of the problem frames and should be chosen when the meaning of the input and output is not of interest and it can be treated as mere symbols.

In his preface, (2001, p xiii) Jackson states ' we are not going to discuss how to elicit requirements' and this concludes the step of elicitation. The Problem Frame approach does not provide any technique for requirements elicitation. In order to apply the method the elicitation step must have first been completed by another approach. The advantage of this lack of technique is that the analyst is free to use their preferred elicitation techniques and still go ahead with the Problem Frame approach in full. The disadvantage is of course that there is no guidance for requirements elicitation so the Problem Frame approach is not a complete approach for the task of Requirements Analysis. The whole task relies on a firm knowledge base of the problem area and so requirements elicitation is a crucial building block for future

work. If the correct information is not gathered, how can the process continue successfully? The step of requirements elicitation must be approached with the idea of problem decomposition in mind so that enough information on dependencies between functions is observed and gathered.

After the information has been elicited, the step of modelling can begin. The aim of this method is to partition the problem into sub-problems, each fitting one of the defined problem frames. Jackson notes that fitting a problem to one of the basic problem frames is often going to appear difficult in the real world as many problems are made up of complex requirements. He states that complexity should be handled by decomposing problems into smaller sections. Jackson (1997, p239) states that 'We structure a complex whole by separating it into parts that can be analysed and considered independently.' It is thought that problems can be repetitively broken down into smaller problems so that they do eventually fit one of the basic problem frames, or a flavour or variant of one.

Rapanotti et al., (2004) highlighted that a disadvantage of the Problem Frames method was the lack of guidance on decomposition of problems into sub-problems and the inability to accommodate change as the real world problem evolves and changes. Jackson (2001, p59) also makes the following comment regarding decomposition, 'The key to this approach is prior knowledge: knowledge of the problem classes; knowledge of what's involved in solving them; and knowledge of the concerns and difficulties that can arise for each problem class.' This requirement of pre-requisite knowledge is therefore a disadvantage for people who are using the method for the first time. It is very specific knowledge that might take a lot of time and effort to build up. However, once this has been achieved, the reuse of the knowledge will be a definite advantage to help deliver accurate, previously tested solutions to new problems.

Each problem frame has a problem frame diagram, the notation and summarised examples are provided in appendix B. The drawing up of these diagrams and showing how the sub-problems interlink is part of the modelling step. The highlighted requirement for business

users to be active in the creation of requirements specification documents for financial systems could be assisted by the existence of diagrammatic representations. As there are set problem frame diagrams this may help the user to eventually become familiar and comfortable with being involved during the problem decomposition and problem frame identification stages. The problem frame diagrams are similar to context diagrams but with extra representations. Context diagrams are already widely used and understood and so the knowledge base is already in place to progress to problem frame diagrams. This can be viewed as an advantage when considering the ease of inexperienced people being able to contribute to the Requirements Analysis process.

Tools do not exist to assist with the method used or creation of the problem diagrams. This could be classed as a disadvantage as tools can speed up a method and provide consistency of application.

The concept of problem complexity has continued to be studied and another area of research has resulted in the problem frame extension idea of the architectural frame, known as the 'Aframe' (Rapanotti et al., 2004). Also, Laney et al., (2004), created the concept of the Composition frame to handle situations where inconsistencies between requirements appear. However, these are beyond the scope of this project and are mentioned only to highlight that further research in this area is on going.

As seen, the Problem Frame approach has been developed to allow it to be more readily applied to real-world problems and handle complexity. The research has also been carried out with the realisation that reuse is a desired and often necessary tool to achieve speed of delivery and confidence in the quality of the newly created solution. Once the problem domain has been matched with a specific problem frame, the solution methods already in existence for that particular problem frame can be discussed, analysed and considered (Jackson, 1995a). This makes the process very systematic and opens up the concept of reuse in the early stages of the development lifecycle that can bring many advantages. As

Robertson and Robertson (1999, p233) state 'If you reuse the requirement, you probably get the design and the code/objects for free. By reusing earlier in the cycle, you get the advantages of the downstream products'. Reuse from the problem domain analysis stage is probably one of the earliest stages that you could start to consider reusing existing components.

Once the sub-problems have been identified and the problem diagrams have been created, the analysis of each frame concern can begin. Each frame has an associated frame concern, that once considered will assure the analyst, and the customer stakeholders, that the requirements modelled pass a correctness test. It is here that the requirement, domain and specification are described in full, each description having its own focus and future use and so staying distinctly in three separate parts. Jackson (2001) states that the ideal for the Problem Frame method would be for each problem frame to have a systematic method to make the analysis and solution identification easy. However, he admits that this level of development has not been reached yet and in the interim, the specific frame concerns do highlight all of the things that each category needs to consider. In the later literature that I have reviewed I have not found any development in this area and so this could be a disadvantage when applying this method, especially for an analyst without experience of the method.

Finally, each sub-problem domain must be analysed to ensure consistency of data and behaviour to take account of how to put the individual solution parts together so that they do not adversely affect the system being built. Jackson (1994) comments that this can be a difficult task where the same domain is defined within two different problem frames, each utilising different characteristics, as the solution composition then has to give that one domain both sets of characteristics. Jackson illustrates this with the example comparison of the V-2 rocket that is assembled with individual components, external skin, structural rods, tank wall, etc., with the Saturn-B moon rocket where the external skin is at the same time a structural component and also the tank wall. The interaction between the sub-problem solutions and the

composition of the whole solution can therefore be a very complex task that requires a lot of consideration by the analysts.

A further aspect of financial environments that has to be considered is the adaptability to changing requirements, due to regulations for example. The Problem Frame method is very categorised and if the change required meant that the sub-problem no longer fit it's basic problem frame then the whole analysis step would have to be reworked to comply with the new problem frame, analysing the new associated frame concern.

Since the concept was first published, Problem Frames have been the subject of academic research and commercial case studies. In 2004 the 1st International Workshop on Advances and Applications of Problem Frames was held allowing all involved in the Problem Frame community to meet and discuss their ideas. As the method becomes more widely acknowledged and used it is intended that advantages and disadvantages will be highlighted and developments made to utilise or over come these respectively.

2.4 Experiences Using KAOS

In 1990 two related projects under the topic of Requirements Analysis were underway in a Belgium University, the University of Namur. One was named KAOS and was based on requirements elicitation, the other was named ICARUS and was based on requirements formalization, i.e. creating a formal specification.

With an underlying concept of eliciting requirements only, the KAOS project focused on goals to be achieved by a system with the intention of guiding analysts when creating a requirements model. The project team felt that 'requirements elicitation should be structured in terms of a model for acquiring requirements models' (Dardenne et al., 1991, p14). The project created the KAOS meta-model for requirements modelling, see appendix F for this

model. By 1993 the KAOS methodology for eliciting, specifying and analysing goals and requirements was in the academic domain (van Lamsweerde, 2000).

The method is based on identifying system goals, goals that *must* be met, and private goals, goals that specific agents might want and so *may* be met. The goals initially identified need to be 'reduced' into leaf goals that can be operationalized by specific agents, preferably single agents.

KAOS is one of many goal-based methods for Requirements Analysis. For a definition of a goal, van Lamsweerde (2001, p250) states that they are 'an objective the system under consideration should achieve.' These methods therefore rely on the goals being known. The requirements elicitation step will need to identify the goals that the system to be built will have to achieve. This is an area which Anton (1996) finds lacking support in some goal-based methods. They do not explain how to identify the goals on which to follow through with. It was with this in mind that Anton and her team developed the Goal Based Requirements Analysis Method (GBRAM) to 'identify, elaborate, refine and organize goals for requirements specification' (Anton, 1996, p136). A full definition of this method is beyond the scope of this project. However, in brief, GBRAM is a method that initially lists useful documents, like flow charts, that can be examined to extract process descriptions and action descriptions that will lead to the identification of goals and the agent that will be responsible for their achievement. The method then leads on to elaboration of the goals by considering them in sub-goal categories and applying their refining techniques until a set of goal schemas have been identified that can then be translated into requirement specifications.

Through the requirements elicitation step, possibly using tools like GBRAM, most goals will therefore be identified at the beginning of the analysis step as they form the basis for the goal-based method. However, many articles agree that there should not be a deadline for identification of all goals. Some, especially goals based on technical detail, might well be discovered later on. Goals raised early in the requirements elicitation stage will also

possibly change as the users redefine the system they have in mind or discover new scenarios they want to be considered.

Goals are often arranged hierarchically, with the overall general goal at the top and its related sub-goals branching out underneath, see appendix E for examples. Again, as the goals are broken down into sub-goals, it is this reduction that helps to handle complexity. Reduction is a top-down process, however, from the literature, it is commonly agreed that it is also good practice to analyse the goals in a bottom-up approach as well to try and check for completeness. Further on the topic of complexity, in one study van Lamsweerde et al. (1995, p197) noted a difficulty of the goal approach regarding goal reduction for complex problem domains. They found that often a goal could be reduced in a number of different ways and deciding which way would prove most effective was difficult to determine. It was only later on in the requirement specification process that difficulties of some reductions arose. It was then a large task to back track and consider an alternative goal breakdown. This issue arose several times during their application of the KAOS method to a meeting scheduler Requirements Analysis task, so it is quite a disadvantage and concern.

The complexity of real world systems needed the concept of goal-based analysis to be developed further. Anton and Potts (1998, p159) state that sub-goals 'may introduce complications and retract simplifying assumptions'. The solution to this was to introduce the concept of 'obstacle' conditions. Obstacles are properties that might hinder the fulfilment of a goal. Methods to highlight obstacles are therefore required as well as a way of transforming goals to mitigate the effect of the obstacles or overcome them completely (van Lamsweerde and Letier, 1998). It is felt that it is better to handle these obstacles in the early goal defining stages rather than mitigate them further into the development life cycle. To identify obstacles, questions should be asked regarding goal dependencies and preconditions. If these are not met the goal will fail and so an obstacle condition has been identified. By considering these situations, *hidden* goals can sometimes be found. An example to illustrate this concept is

provided in section 4.4. There are possible ways of handling the identified obstacles.

- Redefining existing goals
- Adding new goals
- Reconsidering the assignment of goals to different agents

Goal-based methods also have a concept of 'conflict' conditions. These usually arise from two separate stakeholder groups who have slightly conflicting views on the requirements. Although all conflicts must be resolved by the time the solution is presented, conflict conditions should be highlighted and carried along through the elicitation, modelling and initial analysis stages as each might reveal important considerations about the problem domain which might otherwise remain hidden. This concept can be illustrated by an example from Bolchini and Mylopoulos (2003). The problem domain is a museum website and they have two conflicting goals of 'Highlight Munch's unknown work' and 'Highlight Munch's famous works' from different stakeholders. They state that by highlighting this conflict, they helped the designers to consider each goal and thus accommodate both. Being aware of these conditions at the early stages allows the task of elicitation to deliver a richer specification.

These techniques were proposed to be integrated into KAOS so that it can handle complexity and real world cases better. Any goal dependencies need to be highlighted early so that conditions can be determined to allow all goals to be realisable. This acceptance of conflict and obstacle conditions, together with the proposed management of each, enables the goal-based methods to be used in real world cases. The current KAOS method acknowledges both obstacle and conflict conditions and manages them in the ways shown above.

Once goals have been identified they are often classified into different types. Anton and Potts (1998) state that goals can be categorized in different ways in order to gain groups of similar types that can all then be explored in a similar way. They state that one useful way of

classifying goals should be according to the type of target condition desired. These are:

- an improvement of some metric
- the achievement of a desired state
- the avoidance of an undesired state
- the preservation of a desired condition

They also go on to say that goals can be classified according to their subject matter, which they interestingly compare to Jackson's problem frame idea. There are many different classification systems, the use of which is very important for the concept of reuse early in the development life cycle. After the requirements elicitation step has highlighted the goals involved in the problem domain, categorisation techniques will help to speed up the delivery of a solution by reusing previous solutions to similar goals and also by building up a catalogue of considerations or difficulties inherent to specific goal classes. However, which classification system should be used? There does not appear to be any specification, which implies that the user of the method can choose. This is an advantage in that the method is not too prescriptive and different environments can apply their own specific classification system. It can also be seen as a disadvantage in that the most successful classification system might not be obvious for specific environments and therefore the full benefits will not be gained.

In van Lamsweerde (2001a) it is noted that goal refinements are modelled by using AND/OR graphs, see appendix E for examples. AND-refinement links sub-goals together that must *all* be satisfied in order for the goal to be satisfied. OR-refinement links sub-goals where satisfying *any one* of the sub-goals means that the goal is satisfied. This diagrammatic view of the goals will help to involve business users in the task of Requirements Analysis. Goals are often identified by answering the 'what' questions, this is a concept that non-technical people can easily be made comfortable with and contribute to. In van Lamsweerde (2001a) one of the

conclusions drawn highlights an advantage of using KAOS as being that many of the stakeholders are very interested in the goal structures, more so than other models like UML, and this engages an early involvement from a wider audience which in turn produces more reliable software. Also, van Lamsweerde argues, an advantage of using goal over other methods is that more abstract goals can also be highlighted in answer to the 'why' and 'who' questions which are raised during the elicitation and analysis steps. Again, they are then understandable to non-technical people.

Once the leaf goals had been found, van Lamsweerde et al. (1995) noted that the operationalization of these into assignable constraints could prove to be very difficult, especially if there were many agents to choose from. They noted that this could be a disadvantage when applying the method to a new application area. Further, they noted that it was difficult to operationalize more abstract goals as the constraints associated should be more of a concrete action. This is discussed and explained further in Section 4.4.

To further support the use of this method in the real world, a tool called GRAIL was developed to specifically support the KAOS methodology by providing an environment which displays views of the specification in the form of graphics, text, abstract syntax and an object base view as described in Darimont et al. (1997). This tool was developed further alongside industrial projects that were using the methodology, including a telecommunication system and a copyright system. The existence of tools like this should make the application of the method easier and faster to produce reliable documentation.

The KAOS method has been used in various industrial environments at CEDITI (a UCL university spin-off) as reported in van Lamsweerde (2004), although none listed are financial environments. The listed advantages of using the KAOS method include the extraction of robust requirements, the built-in vertical traceability (from business objectives through to technical requirements), precise specifications and architectural design choices (van Lamsweerde, 2004). This visibility allows the business users as well as the technical

developers to engage in meaningful discussion throughout the Requirements Analysis stages. A two way correspondence is very useful to keep goals on track and keep communication lines open in case change is required mid project. This visibility will also be useful for future times when the development falls into the classification of a legacy system. It will then enable future developers to have a clear view of the functionality and reason's behind it.

However, I must note that, being one of the creators of the KAOS method, Lamsweerde is considered a master of such and therefore the ease with which he applies it may not be reproducible by all developers and analysts. I must therefore consider that many of the advantages might not be realisable until mastery is achieved.

2.5 Experiences Using Combinations of Methods

Each individual method has been discussed above, however, in the initial aim it was acknowledged that a combination of methods might prove to be the best solution for requirements specification of financial systems. Research in the area of combining techniques and methods has also been reviewed and some instances are highlighted below.

The KAOS method was applied to a meeting scheduler specification exercise in van Lamsweerde et al (1995, p203) and their conclusion was that where KAOS was lacking, in this example in the ability to accurately assign an appropriate agent, other known techniques, namely scenario-based and viewpoint-based techniques, complemented the goal-based technique to produce a complete and consistent specification. It was this 'hybrid strategy' that they felt produced the best results.

In a further study based on gathering Web site requirements, although KAOS was not the chosen goal-based method, it was highlighted that goal-based methods prove to be very good when in the early stages of gathering requirements that are unstructured and ill-defined. This

is a common aspect when designing web sites as the user often does not know exactly what information they would like to receive. It was thought that goal-based techniques should be used in the early stages of Requirements Analysis and task-oriented techniques used in the later stages (Bolchini and Mylopoulos, 2003).

Case studies have also been carried out to apply the Problem Frame method on its own as well as to integrate it with other Requirement Analysis methods. An example of a trial integration is provided by Bleistein et al. (2004) where a Problem Frame approach was integrated with goal-oriented modelling techniques and applied to an e-business environment, Seven-Eleven Japan's e-business system, as a case study. Both advantages and disadvantages of this integration were noted due to the different methods complimenting each other in some areas and then being irreconcilable in others. An example of where they were complimentary is when the initial modelling using problem diagrams was then a useful boundary to express the behavioural aspects uncovered by the use of goals. An example of where the two were irreconcilable is when the Goal-Oriented Requirements Language's (GRL) context description could not be mapped to the context diagrams of the Problem Frame approach. However, the overall feeling was that 'the integration of the Problem Frames approach, goal-oriented modelling techniques, and business process modeling may offer promise as a requirements engineering tool for e-business systems' (Bleistein et al., 2004, p416).

It does therefore seem plausible that the best results might be achieved when mixing two or more methods.

2.6 Summary

Through my literature search I have gained a full understanding of the background and details of the current underlying problems with the task of Requirements Analysis and specifically

the characteristics that are associated with financial environments. This then provided the background for my literature search of the two chosen methods to investigate how they have been analysed and applied to date. The literature review for the two methods has highlighted the techniques that each uses for Requirements Analysis as well as the advantages and disadvantages that have been found whilst the method has been applied both in academic and industrial environments.

The aim of my literature search was to find the work of the main researchers in each area under consideration and to understand what had been achieved so far. It has provided an insight into the application of the two methods in non-financial environments as, just as I anticipated, only a limited amount of literature was available relating to a financial environment. However, this aspect of my work has supplied information that I will now take forward for my comparison of the two techniques in a financial environment.

These methods span across the steps of elicitation, modelling and analysis of the requirements and a summary of how each handles these tasks as well as how they address the general and financial environment specific characteristics has been summarised in Table 2.2. These considerations will be taken forward in to the case study to see whether they are still valid when the methods are applied in a financial environment.

Task / Characteristic	Problem Frames	KAOS
Elicitation	This method does not have a specific technique for elicitation and as such the information may or may not be presented in an organised manner in order to begin modelling from.	This task is carried out by deriving system and private goals from the problem domain. There are techniques to help with this process but it is documented as being a difficult task. However, because of the way in which the data is being gathered it does produce well structured information.
Modelling	The initial model is a Context diagram, followed by a problem diagram and further sub-problem diagrams. MS word can produce them without too much difficulty.	AND/OR graphs are used to model the problem domain. They are clear, concise diagrams and do not have many symbols or notations to learn. MS word can produce them without difficulty.
Analysis	Specific frame concerns and general concerns are specified to analyse the sub-problems fully.	Goal reduction and operationalization techniques are used to analyse the requirements that have been elicited.
Time spent gaining knowledge about the problem domain	Knowledge of the problem domain is encouraged but no specific task requires this. The method starts once the initial requirements have been presented.	KAOS encourages a lot of the elicitation time to be spent amongst the user stakeholders gaining knowledge of the problem domain in order to establish the goals.
Models created?	The problem diagrams model the relationships and behaviours.	The AND/OR graphs model the information and relationships to quite a good extent.

Problem is divided into manageable parts?	Problem is partitioned into sub-problems until recognisable problems are found.	Each goal is worked on individually and so the whole problem is divided into manageable parts.
Information gradually increases in detail?	The frame concerns build up the detail from an initial high level view.	The method leads into gathering more detailed information and behavioural links after initially starting quite high level.
The principles behind the method are easy to learn?	The principles behind the method are quite difficult to learn and understand and until some base level is gained the method is not easy to apply. It really needs mentoring until some level of mastery is gained.	The concept of goals is quite widely used by other methods so there is potential for previous experience. However, the analysis techniques require practice and possibly mentoring.
Complexity	The decomposition handles complexity but there are still concerns about the problem frames modelling real problems. The sub-problem approach does allow traceability from problem to solution.	Obstacle and conflict conditions have been developed to help handle real world situations. Also, AND/OR graphs match 'what', 'how', 'why' and 'who' questions to provide vertical traceability for the technical and non-technical stakeholders to allow all to contribute.
Regulations	Changing requirements have to be catered for on longer projects and the difficulty to adapt the models would depend on whether the change meant that the frame chosen had to change. If not, then the analysis could just be completed again to ensure consistency.	Changing requirements have to be catered for on longer projects and some discussions on KAOS show that altering the goal reduction is not always an easy task.

Competitive edge	Delivery of projects could be made faster once the method is mastered and the solutions to each frame have been catalogued for reuse. This method lends itself well to reuse.	Delivery of projects can be made faster by reusing development documents and past experience through the use of a goal classification system. Also tools like GRAIL will help lower the project delivery time.
Legacy systems	The problem diagrams do produce clear documentation explaining the partitions of the problem and the interaction between problems would also be documented. This should aid the replacement of specific systems by function if needed.	KAOS does not specify any development language or specific notation, other than the AND/OR graph so any documentation style can be used. The AND/OR graphs do produce clear, reliable and vertically traceable documentation to accompany the final system though.

Table 2.2: Comparison of Problem Frames and KAOS from the Literature Review

Chapter 3 Research Methods Used

This dissertation falls into the categories of analytical, empirical and evaluative. I have initially analysed the available literature so that I could apply the knowledge gained to a case study and finally evaluate the results to draw conclusions and present recommendations and guidelines. Appropriate research methods for each section of the work were therefore required.

3.1 Defining the Case Study

From the literature that I used to review the two methods, the research method that had often been used was that of a case study. In Laney et al. (2004) an introduction to Problem Frames was given, citing relevant articles from their literature search followed by a case study where they applied the method when gathering requirements for the control of a sluice gate. Also, in Anton (1996) the introduction provides results and citations from the literature search followed by a case study using their new technique, GBRAM, to analyse the goals for a Career Track Training System. Therefore, in order to apply the knowledge gained from the literature to a financial environment, and accommodating my limited time resource, I decided to use a single-case study. My objective was to apply both the Problem Frame approach and the KAOS method to the same problem domain to produce a requirement excerpt document demonstrating the two methods. The document will not be a full specification of the whole problem requirements but a summary / overview of the Requirements Analysis steps carried out so that assessments and evaluations can be made. The approach of a single-case study is limiting and may limit the effect of any conclusions that I have drawn. Also, this meant that one method followed the other using the same subject so that could also have limited the effect of my results and also may have caused bias. However, it has allowed me to gather

detailed, descriptive evidence from a financial environment that I have then used to make generalisations from and so I believe it to be valid research. Yin (1994, p4) highlights two famous case studies that are single-case studies and yet very well quoted and valued because of their 'generalizability'. For reference, these are Whyte's 'Street Corner Society' and Allison's 'Essence of Decision: Explaining the Cuban Missile Crisis'. I have aimed to also make my results valid in a general sense so that my research will be valuable to a wider audience. When I applied each method, I reviewed against the framework of tasks and characteristics as illustrated in table 2.2. This allowed me to gather comparable feedback of each method.

3.2 Review and Evaluation

In both Laney et al. (2004) and Anton (1996), reference is made by the authors to the discussion of the findings with a team of colleagues. My alternative to this was to hold the design / analyst and system tester review meetings. When I had created the requirement excerpt document ready for review I arranged two separate reviews with teams of designers/ analysts and system testers respectively. These teams consisted of a small group of my colleagues who carry out design/analysis and system test activities daily within a financial environment. Each meeting had a review of the requirements documentation to see whether they were complete and unambiguous to allow the next stage of the development process to begin i.e. design and system test. The designers' / analysts' meeting then had a discussion of each method.

The aim of presenting the requirement excerpt document to the designers / analysts and the system testers is to gather further input for my comparison of the methods. This required careful planning as unbiased, evaluative, data is required so that it can contribute to my overall guidelines and conclusions. The meeting required guidance by myself, referencing the

documented advantages and disadvantages found from my literature search, but I was conscious of the need to obtain a balance between providing the information to set the meeting and biasing the outcome. In preparation I referred to 'Case Study Research: Design and Methods' (Yin, 1994) for guidance on good practice. I did not produce a transcript of the meetings but instead carried out content analysis on the proceedings. I also referred to 'Methods of Text and Discourse Analysis: In Search of Meaning.' (Titscher et al., 2003) to understand the method and concerns behind this research technique.

To ensure that the review meeting participants had every opportunity to express their views I also prepared a questionnaire to gather any specific feedback that I thought would help me to draw my conclusions. The questionnaires were designed to be completed immediately after the review meetings by the participants themselves. This strategy was to encourage feedback on any areas that the participants did not feel confident enough to address during the review whilst ensuring that I was on hand to be questioned if any uncertainty arose. Whilst carrying out the case study I made notes of areas that I wanted opinions on and this provided the basis for the questionnaire. As only a small number of questionnaires were going to be completed they cannot be used for any useful statistical analysis. However, I believe that they will assist me to assess the feedback on the two methods and to record accurately the content of the review meetings.

Before I conducted the reviews I performed a walk through of my script to ensure that I would gather the information that I needed in a format that would be useful for my conclusions. I also requested a colleague to complete a copy of the questionnaire before handing out to the teams so that I could correct any areas that did not gather the required data or might have been interpreted as biased. My learning points from the pilot questionnaire were to make the instructions clearer by amending the scales 1 –5 to correspond to each other i.e. 1 represents easy and useful, both good things, and 5 represents difficult and not useful, both undesirable qualities, see appendix H for details. A further amendment made as a result

of the pilot was to word question 8 in such a way that the reviewer did not have a choice to not comment. This then ensured that all reviewers expressed an opinion.

By following the examples of proven research methods and taking guidance from research method reference books I believe that my chosen research methods were appropriate and effective in gathering good quality data to answer my research questions, although I acknowledge the limitations of a single-case study and any resulting effect of bias.

Chapter 4 The Case Study

This chapter documents how the two methods were applied to the case study and how they handled the steps of elicitation, modelling and analysis whilst carrying out Requirements Analysis within a financial environment. For reference, the definition of each step is provided in Chapter 1. Initially overviews of the financial environment and the problem used for the case study are provided.

4.1 The Financial Environment for this Study

The financial system used for the case study is an in-house built, globally used, commercial banking system of an international financial company. Its functionality includes basic banking, payments, central bank clearing, import and export business and has varied front end interfaces including web based, interactive voice recognition telephone services, call centre technology and the back office VDU screens. Some areas have complex requirements like the imports and exports modules that adhere to the laws surrounding that type of business or the various taxation laws around the world. Also, there are many legacy systems that are still relied upon and often form a chain in a long link of processes. Based on the literature review of financial environment characteristics, this system is a good base for a case study as it exhibits many of the highlighted characteristics.

4.2 The Export Cash Cover Problem

This problem relates to an existing imports / exports financial system that is used to record and control import and export trading business. Currently, when an export documentary credit (export DC) is recorded there is no integral way of associating it to a Cash Cover payment

that might be made as security for adding confirmation to the export DC or to enable the issuance of an unconfirmed export DC. This means that there is no way of automatically linking a payment to an associated export DC. New functionality has been requested to allow this to be automated and incorporated into the existing system, supplying appropriate internal and external reporting and to take account of currency rate changes where cross currency payments are used, appendix A has further details.

4.3 Steps Taken to Apply the Problem Frame Method

The following steps to apply the Problem Frame method were taken from the literature that I had studied. I had not applied this method before nor experienced its application in any way prior to this case study. The chronological order of tasks is as follows.

- Carry out the step of requirements elicitation by my usual approach of reading user documentation, observing users, talking to users and analysing the existing system because the Problem Frame method does not cater for elicitation.

- Create a context diagram to show the boundary of the problem domain. This is noted by Jackson as a useful prerequisite step to lead into the specific Problem Frame method diagrams.

- Create a problem diagram.

- Decompose into sub-problems and create sub-problem diagrams.

- Fit each sub-problem to a known problem frame.

- Analyse each sub-problem with regards to its specific frame concern and the general concerns highlighted.

- Produce the requirement, domain and specification descriptions for each sub-problem.

- Compose the whole solution from the sub-problem solutions.

Whilst carrying out the above tasks I evaluated the method, a discussion of my experience and my evaluation follows.

Elicitation within the Problem Frame Method

In order to apply the Problem Frames method the elicitation step needed to first be completed by an approach of my own. I will therefore not comment on this step other than to say that my usual method did generate a lot of information and I stopped once I felt I had a good view of the high level problem.

Modelling within the Problem Frame Method

The first step to modelling the information elicited about the problem domain is to create a context diagram. The context diagram bounds the problem domain, shows all interfaces and also contains the solution. It does not however, explicitly show the requirement. I am already experienced in creating context diagrams so this was a familiar starting point, an advantage when trying the method for the first time, see appendix A for details.

The next step is to produce a model called a problem diagram. This is developed from the context diagram and shows the problem requirement. It provides more information about the interfaces by introducing a more advanced notation to that of context diagrams and also explicitly adds the requirement and a requirement reference. Jackson (2001) has documented the building of problem diagrams discussing how to model / handle a variety of scenarios like external databases, conceptual domains, connection domains and machine domains. After reading the examples of the process it was possible to begin to create problem diagrams using the prescribed notation, see appendix B for details. The requirement is present on the

problem diagram as a description contained within a dashed oval. It is connected to a domain by either a dashed line or a dashed arrow, the latter being a constraining requirement reference that prescribes a relationship or behaviour of that domain rather than just referring to it. In creating the diagrams I needed to loop back to the step of elicitation a couple of times before feeling confident that the problem diagram was an accurate reflection of the problem domain. However, collecting the required information for this method is something that will improve with practice and is of course dependent on the complexity of the problem domain.. This method of modelling the system would be of help in years to come when the system is considered as a legacy system and clear documentation is required so the extra effort to ensure all detail is gathered is very worth while.

Jackson (2001) describes his idea of phenomena as being individuals – entities, events or values – or relations – roles, states or truths. Shared phenomena, once identified, are represented on the problem diagram by an identifier and below the diagram are defined and annotated with the identifier of the domain that controls the phenomena. The interface line connects all domains that share the phenomena, if this is greater than two then they are joined by a hyperarc.

Specifics can be seen in appendix B, however, generally I had to decide whether identified domains were separate domains or could be considered as one single domain. On considering the future decomposition of the problem both choices were clearly going to fit the transformation problem frame but had different shared phenomena so a decision to separate them and see what the process revealed further down the line was made. I also chose to include the external database as it is an integral part of the problem domain, and clearly the final solution. However I chose to not include the database's shared phenomena with the identified external system as this appeared remote to the problem. Finally, on the phenomena, I had to choose whether to ignore or not the connection domain, to decide whether it is relevant to my problem domain.

The problem diagram notation is therefore quite well developed and the end diagram contains a lot of information about the problem to be analysed. It helps to ensure that the elicitation of information has been completed sufficiently. I believe this to be a big advantage of the method, once the model is completed it is a very strong basis to progress to the analysis step. I have found that the problem diagram modelling technique is relevant for a financial environment, and although it reflects a simplified version of the problem, it is useful to have before commencing the analysis.

Analysis within the Problem Frame Method

Problem complexity requires very involved looking problem diagrams and financial environments do have complex problem domains. This method addresses complexity with decomposition into sub-problems. The idea is to break the problem domain down into overlapping sub-problems that each fit a predefined simpler problem frame class that is well documented with familiar techniques to find their solution. Jackson (2001) highlights that the decomposition should follow some basic guidelines and produce sub-problems that are:

- Complete problems in their own right.
- Projections of the full problem and not a hierarchical breakdown or partitions, see Figure 4.1.
- Concurrent regarding interaction between each other.
- Composite problem frames with a familiar standard decomposition into elementary sub-problems.

The step of analysis begins with decomposing the problem diagram into sub-problems: showing the sub-problem machine; domains directly involved; and the shared phenomena. The sub-problem diagram uses the same notation as the main problem diagrams.

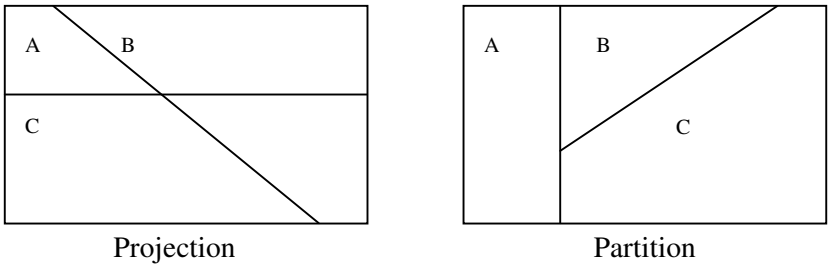


Figure 4.1 Projection v Partition

As noted in the literature review, knowledge of the method and the problem frames is really required in order to apply the method successfully. This is something that I did not have, however I do have experience in the actual environment and this is documented as being of help. Therefore when decomposing into sub-problems I found that it was achievable with reference to the instructions and examples in Jackson (2001). After consideration I managed to break my case study problem into six different sub-problems covering the following problem frame classifications: Information display; Commanded behaviour; Required behaviour and Transformation. The next step was to then draw the sub-problem diagrams, a sample of these can be seen in appendix C.

After decomposing into sub-problems associated with set problem frames, each then had to be analysed to ensure that the requirements were met and complete. This is addressed by the concept of frame concerns that are attached to each frame type and provide the main issues to be considered for specific classifications of problem. This use of classification at such a low level is an advantage that helps speed up the delivery of the analysis step, ensuring that any erroneous frame identification or missing information about the problem domain is highlighted and dealt with. As well as the frame specific concern to address, Jackson (2001) highlights some additional concerns that should be reviewed regardless of the problem frame that the problem domain matches. These are:

- Overrun, the ability to respond to each external event.
- Initialisation, the initial state that the machine has to cope with.
- Reliability, the failure to fit a baseline requirement description.
- Identities, to ensure distinct identities can be identified, if necessary, where multiple instances occur and are not related to each other.
- Completeness, do the problem requirement, problem domain and machine description provide the full picture without ambiguities.

The next step for my case study was therefore to analyse each of my sub-problems using the relevant frame concerns and then the general highlighted concerns. This leads to the production of a requirement, domain and specification description for each sub-problem where the specification combined with the domain description is shown to fulfil the requirement. Appendix G shows an example based on the transformation problem frame. When studying the transformation problem frame I noted that the example in Jackson (2001) is a simple report and that in reality more controlled page breaks would be required, also more calculations, deeper detail and formatting would be essential. However, I found that this complexity difference did not affect the three descriptions in terms of how the information was gathered, it just meant that it was more complex to describe. Regarding the general concerns, the reliability concern did highlight the need to define the action to take if some required information could not be found whilst processing. This issue had not arisen before and it provided an opportunity to discuss the area again with the user stakeholders. In conclusion, the problem frame did fit the sub-problem perfectly so it was the correct problem frame. The frame concern was very relevant and by considering the 'specified traversal rules' I believe that it provided a good specification.

After all of the sub-problems had been analysed in this way the different solutions were

assembled to create the whole requirement specification document. In this case study I did not find that any sub-problems adversely affected any other. The 'export CC (Cash Cover) clerk input' sub-problem triggered the 'export CC accounting entries' solution into action so there was a dependency there. Also the existing accounting database was involved in all sub-problems so this meant that the sub-problem solutions had to be considered together to ensure that the database table indexes were not contradictory and did provide an efficient final system, see appendix C for details. Each sub-problem contributed to the design of the new export cash cover database tables.

In this application of the Problem Frame method I did not have time to see how the method would adapt to a change in requirements. As discussed in chapter 2, this would depend on whether the sub-problems were still consistent with their allocated problem frame.

4.4 Steps Taken to Apply the KAOS Method

The steps taken to apply the KAOS methodology were identified from the description made by Darimont et al. (1997, p612) and the strategy described by Dardenne et al. (1991). I had not applied this method before nor had I experienced its application first hand in any way prior to this case study. The chronological order of tasks is as follows:

- Construct an initial list of goals related to the problem domain.
- Repetitively refine and identify goals until leaf goals were found that could be identified to individual agents. This task is called goal reduction.
- Build a goal structure model, an AND/OR graph, showing the reduction from parent goals through to leaf goals.
- Identify possible conflicts between system and private goals of agents.

- Operationalize the leaf goals by identifying objects and actions that can be associated to the agents.

- Specify attributes and links for objects and actions involved.

- Assign a specific agent to each task.

Elicitation within the KAOS Method

This step is to identify goals through elicitation tasks, although the KAOS method does not stipulate or recommend any goal identifying techniques. The advantage of this is freedom of choice to use techniques that are familiar and well tested within the specific environment. The disadvantage is that if the analyst is not experienced in identifying goals, it is a daunting task to embark on without any prescribed direction.

My previous experience did not include any goal-based methods and so the task of identifying goals was new to me. After discussing the requirements with the business user and reviewing the existing procedural documents and the problem domain I managed to identify some system goals that I believe set out the requirements of the business users. At this stage, the only private goal that I identified related to the working hours of the system users. This could have been because of my lack of experience or it could have been because they were not as applicable in this abstract area where most goals are system goals, appendix D lists the goals.

The reduction of these goals was my next task as each goal had to be reduced to a leaf goal that could ultimately be assigned to a single agent. There is a lot of literature regarding this area and by using the examples in Dardenne et al. (1991), van Lamsweerde et al. (1995) and van Lamsweerde (2001), amongst others, I reduced the goals into branches with leaf goals as can be seen in appendix E. This process was carried out by asking ‘how’ questions of the initial goals identified, i.e. a top-down approach. Asking ‘why’ questions then checked that

the top goals had indeed been identified in the first place.

Example: In the 'Have an accessible list of issuing banks for which CC is required' high-level goal there is a leaf goal of 'Know user category'. Reversing this, if we initially asked, 'why do we need to know the user category?' the answer would reveal that there are different classifications of user, each with their own responsibilities, knowledge and expertise. Some categories are responsible for maintaining the information and some have a need to be able to read the information. This then checks that the branching is correct and if a third reason was found it would have highlighted a need to review the goal reduction.

One decision to be made at this early stage was the way in which the reduction was carried out. From the literature, possible approaches to reduction are:

- Agent-driven, to reduce into agent prescribed goals.
- Case-driven, to reduce into specific cases ranging from normal to exceptional.
- Time-driven, to reduce into successive goals that should happen in order.

I found myself often considering the agent-driven reduction approach as this must have been the clearest route in my mind. However, whether this was the best approach is hard to say and whether an alternative approach would have made a large difference to the requirements specification is also hard to say without actually carrying out the process of reduction again. With this in mind, together with the knowledge that the literature review revealed the need to be able to cater for functional changes at short notice, it would have been a useful exercise to perform the reduction again with a new piece of information to see how the method handled this. Unfortunately time did not allow for this task.

As I carried out the task of reduction I also spent time looking for conflict and obstacle conditions as these can be used to reveal further goals and make the requirements more robust

and complete. An example of an obstacle condition is below, see appendix E for details, however, I did not locate any conflict conditions.

Example: An obstacle condition can be found by asking the question, 'what is the effect on the goal 'Calculate new value of cross currency deals' if the current exchange rates are not known?' If the most up to date exchange rates have not been received then there is no benefit from calculating the new values as they will not have altered. This highlights a time dependency between leaf goals, as the leaf goal 'Receive treasury feed of exchange rates' must happen prior to this one. To address this obstacle condition the leaf goal 'Receive treasury feed of exchange rates' could be redefined to 'Fetch treasury feed of exchange rates' or it could be assigned the Export supervisor agent to take responsibility or introduce new goals to cater for this occurrence like, 'Issue warning report that new values have not been calculated'. Which ever way is decided, the specification will be richer for having considered this obstacle.

Overall, it was quite a difficult task to reduce the goals down as the examples in the literature were of very different environments and none were from financial environments. As there are often different ways to reduce goal structures it was difficult to be confident that the correct leaf goals had been identified. Also, once a structure had been put in place it was quite difficult to start from the beginning and try to reduce in a different way just to test the approach as the first attempt always biased the secondary attempt. I believe that this task requires a lot of experience and I am lacking in this area at the moment.

Modelling within the KAOS Method

The goals are diagrammatically illustrated by AND/OR graphs that are hierarchical tree diagrams highlighting the depth of detail and the origin of each end goal. Appendix E shows the AND/OR graphs created during the case study. When assigning agents to the leaf goals I found it difficult to draw away from the Cash Cover System (CCS), which is the new system

that will result from this requirements specification. The financial environment proved to be very different to the more physical environments like lift systems in Dardenne et al., (1991). It was hard to draw away from the agent being the system itself as most financial environments are highly automated environments that often operate without any human contact. Also, financial systems have many functions that happen simultaneously and independent of each other so it was sometimes also difficult to consider time-driven reduction approaches or sequences of events. However, the resulting AND/OR graphs are clear diagrams that do model the separate aspects of the system and would satisfy the need for clear diagrams as highlighted by the legacy system characteristic.

Unfortunately I did not have the resource to trial the GRAIL tool and as this was my first attempt at applying the method I did not have any catalogued resources to reuse and hence make the method faster and easier. This case study cannot therefore comment on this aspect of KAOS and it might mean that one of the advantages of this method has not been highlighted.

Analysis within the KAOS Method

The first task of analysis is to begin to 'operationalize' (a KAOS term) the leaf goals into constraints. This step is where knowledge of the environment is very beneficial as each goal had to be considered to determine how it could be enforced through the use of state transitions. The literature had implied that this stage could be difficult if there were many agents involved. When carrying out this task I found that as only four agents had been identified and I know the environment well, the assigning of the constraints was not too difficult a task, once the state transitions had been decided upon. For the following examples, see appendix E for details.

Example: Leaf goal 'List displayed' can be operationalized by the two constraints 'List requested' and 'List provided' with the first constraint assigned to either the Export clerk, EC,

or the Export supervisor, ES, and the latter constraint being assigned to the CC system, CCS.

However, as also highlighted in the literature, sometimes identifying the state transitions for particular leaf goals proved to be a difficult task. The following example is one that I found difficult to break down into constraints, however as highlighted above, the obstacle condition has shown the time dependency for this goal so it is an area that did require further consideration.

Example: Leaf goal 'Calculate new value of cross currency deals' can be operationalized by the constraints 'Deal currencies and current values retrieved'; 'Latest currency exchange rates known'; 'Calculation performed'; 'New value recorded for deal' with all of the constraint assigned to the CC system (CCS).

In carrying out the analysis step of operationalization I tended to wonder whether the agent CCS should have been broken down into further more distinct agents so that more specific sections of the CC system could have been grouped and analysed. This type of knowledge I believe would be gained with experience, perhaps compiling a catalogue of experiences that would aid the agent identification in future applications.

The next analysis task was to assign attributes and links. From the literature, some examples of attributes that goals can have are names, priority and state although it was difficult to actually find examples that explained the specification level in this much detail. A goal link can exist between another goal or another element of the requirement model like an agent or an operation. Examples of attributes can bee seen in appendix G, an example of a goal link follows.

Example: As previously highlighted by the obstacle condition example, the goals 'Calculate new value of cross currency deals' and 'Receive treasury feed of exchange rates' are linked by the sequence in which they must happen, the constraints associated with each are linked

and so there is said to be a goal to goal link between these two.

The final stage of the analysis is to associate one specific agent for the performance of each action. Dardenne et al., (1991) specify that the agent chosen for a particular action should have ability, responsibility and motivation for that action. This rule was adhered to when assigning the agents in the case study, however, as previously noted, maybe the CCS agent should have been broken down further into more specific agents.

4.5 Summary

The application of the Problem Frame and KAOS methods to my financial environment case study proved challenging due to lack of experience with the methods. I accept that the final applications may therefore not be quite correct and this could limit the effect of my conclusions. However, I believe that the requirements specification excerpt produced as a result is valid and exhibits key features of each method that can now be reviewed and progressed to produce some applicable guidelines for financial environments. I also believe that my experience gained during both applications allows me to draw conclusions and generalisations about the two approaches in order to use my experience as input to the guidelines.

My overall experience of applying these two methods has highlighted some advantages and disadvantages for each method. In the same format of table 2.2, my experience has been summarised in table 4.1 below.

Task / Characteristic	Problem Frames	KAOS
Elicitation	I had to reiterate this step a few times. The advantage was that it kept the user stakeholders in touch with the process. The disadvantage could be if the user stakeholders were not accessible after the initial elicitation task.	Eliciting system and private goals was a new way of thinking for me. I found it difficult to find private goals, perhaps they are less relevant in financial environments?
Modelling	The problem diagrams contain a lot of information and the notation does need to be learnt. However, after the initial effort to learn them the diagram is a good base to continue analysis from..	AND/OR graphs were quite easy to create once the analysis to reduce the goals had taken place. The diagrams can be built up to represent link and relationship information.
Analysis	Easy to break into sub-problems as knowing the defined problem frames means that you know what you are looking for. The frame concerns give reassurance that the correct breakdown has been achieved and they also help to prove that all information has been elicited.	Goal reduction was difficult as there are no guidelines as to what the final breakdown should look like. Operationalization is also quite difficult for complex goals.
Time spent gaining knowledge about the problem domain	I had to learn a lot about the problem domain in order to apply the frame concerns.	I had to learn a lot about the problem domain in order to elicit the initial goals.
Models created?	The final set of problem and sub-problem diagrams provide a lot of information.	The AND/OR graphs do illustrate the goal breakdown well.

Problem is divided into manageable parts?	As the final sub-problems have to fit one of the simple problem frames they are all of a very manageable parts.	The leaf goals varied in difficulty from some being very manageable to some being quite complex.
Information gradually increases in detail?	The frame concerns break the sub-problems down into three detailed descriptions, requirement, domain and specification.	The final leaf-goal constraints lead into a detailed breakdown of information.
The principles behind the method are easy to learn?	It was quite time consuming to learn the method prior to being able to begin. However, after this initial stage the method became easier to apply. The terminology used in this method is quite difficult to become familiar with.	It was quite fast to start with this method as knowing a little initially allowed me to begin. However, time had to keep being spent learning the skill of the proceeding stages.
Complexity	Complexity was handled as the sub-problems all fitted the specified simple problem frames. However, I believe that complexity of some problems could make the composition of the final solution difficult.	Tackling each leaf-goal separately does handle complexity, however, the leaf-goals do vary in complexity themselves.
Regulations	Changing requirements were not tested in the case study so I cannot comment.	Changing requirements were not tested in the case study so I cannot comment.

Competitive edge	Within a financial environment I can foresee that it would be possible to catalogue the problem frame solutions and hence speed up the delivery of projects.	I did not test the GRAIL tool so I cannot comment. As goal reduction can take any direction I believe that it might be difficult to create a catalogue of model solutions to speed up the method.
Legacy systems	The documentation produced is very clear and would allow sections of the system to be amended individually at a future date.	AND/OR graphs produce documentation that will enable sections of the system to be amended individually at a future date.

Table 4.1: Comparison of Problem Frames and KAOS After Application

My conclusion is that these two methods, or possibly hybrids of each, are applicable to financial environments. My original aim to create recommendations or guidelines to advise when each method, or parts of each method, can be used to the best advantage in a financial environment can therefore progress as planned.

Chapter 5 The Review and Evaluation Process

This section documents the review meetings that were held. Both of the review meetings had a brief overview of each method using the examples in appendices B, C and E. Then there was a review of the requirements documentation excerpt, appendix G, to see whether they were complete and unambiguous to allow the next stage of the development process to begin. For the design team there was a second element to the discussion to review each method considering similar evaluation conditions as chapters 2 and 4. The reviews could not reflect on all aspects highlighted by tables 2.2 and 4.1 as the reviewers were inexperienced in both of the Requirements Analysis methods and the time constraints did not allow for any tuition to enable them to review in depth.

For the excerpts I decided to choose two separate parts of the system so that the reviewers could not remember requirements from one method that might bias the review of the second method. The requirements specification for the whole problem could not be reviewed due to time constraints. I acknowledge therefore that this limits the review of the completed method. To complete the evaluation each reviewer was asked to complete the questionnaire contained in appendix H.

5.1 Results of the Design Team Review Meeting

This review meeting consisted of four experienced analysts / designers who are currently working in development teams in a financial environment. They all had experience in preparing functional requirements documentation as well as design documentation. None had heard of either method although some had experience of Jackson's earlier analysis work. There was no overall preference for a method and I noticed that individuals tended to relate to the method that they associated most with their current method of Requirements Analysis.

Using the methods to design a solution

An important consideration for the designers was that they could rely on the requirement specification and not have too many future changes. Some felt that the Problem Frame method could take longer to produce the requirement documentation as the problem frames had to be identified after the initial elicitation had taken place, but once produced the 'first-cut' of the requirements would be more accurate with only cosmetic changes occurring. Whereas others felt that knowledge of the problem domain had to be greater for the KAOS method to ensure that the correct goals were highlighted and therefore this method would produce a more accurate specification.

All reviewers saw the benefit of the basic problem frames and liked the idea of this classification to catalogue experience. They all saw the KAOS method as being harder to catalogue for future use to speed up the development process and reuse past experience. There was also a feeling that the initial steps for the KAOS method were less structured and the concrete idea of basic problem frames in comparison was seen as a strong point.

The logical flow of the requirements was seen in both methods and during the discussion the problem frame projections and goal hierarchy structures were both seen as traceable and useful ways to break the problem domain into manageable pieces. However, the questionnaire results were slightly in favour of the KAOS method for tracing requirements and hierarchy structure.

The reviewers all agreed that the problem frame diagrams and terminology were an initial barrier and not representative of usual language. The opposite was agreed of KAOS, use of terms like 'obstacle conditions' were seen as good terminology. A further comment on the problem diagrams was that they do not show the logical flow of the processes, the descriptions have to also be read to ascertain the controlling domain. It was agreed that this was not the reviewers' usual style of diagram and so initially the KAOS AND/OR graphs

were more accessible. The ease of interpreting the AND/OR graphs was confirmed by the questionnaire result although no extreme rating (5 represents difficult) was given to the problem diagrams.

Using the methods for Requirements Analysis

All reviewers had many years experience in their financial areas and thought that this would allow them to use the first steps of the KAOS method without any specific guidance. All agreed that tuition in the Problem Frame method as a whole and the KAOS method's analysis techniques would be a necessity. Once this had been gained, all agreed that both methods would be suitable for their financial environments but until they had tried them for themselves it was difficult to comment. One reviewer felt that the KAOS method might not handle complexity as well as Problem Frames as the goal structure was seen as less structured. Others felt that they liked the freedom that the KAOS, goal-based method, allowed.

Conclusion

Two general comments were made that are worthy of consideration: (i) that all terminology needs time to be learnt and become familiar with so this should not be an overriding issue and (ii) that elicitation is very difficult as you could be asking the right question of the right person, but they just might not realise exactly what their problem is until they see a solution that they can then relate to and comment on. With these comments in mind and the above discussion points it can be concluded that neither method was seen as being better than the other overall, but that each had positive aspects and each had areas for further thought.

Thoroughly gathered requirements were deemed necessary to avoid design changes in the future and both methods were judged positively on this aspect. The structure of breaking into sizeable pieces and the diagrammatic representations needed different amounts of tuition and experience but as a long-term consideration these were ruled out as deciding factors. Three out of the four reviewers felt that a formalised structure as thorough as either of these two

methods would create a positive Requirements Analysis approach within their financial environments. However, one felt that a set structure would be restrictive and would prefer freedom of method to suit the situations as they arise.

5.2 Results of the System Tester Review Meeting

This review meeting consisted of four experienced system testers who are currently working in a financial environment. Each had attended and qualified from the ISEB testing courses and regularly prepare test plans for testing in a financial environment. Again, none had heard of either method but they were familiar with and used to using context diagrams.

In this review there was an agreement that they preferred the structure of the Problem Frame method and could not imagine the KAOS AND/OR graphs handling the complexity of their functional requirements in practice. Although my examples broke down into leaf goals they felt that in practice they would need a diagram with a further breakdown of detail. They also agreed that they liked the strong classification structure of the Problem Frame method and felt that it would be a clear way to categorise their test cases so that, as experience grew, they would be able to prepare faster by reusing previous work.

On the analysis side, they liked the idea of the functional specification highlighting obstacles, scenarios, pre-conditions and post-conditions. They felt that these should be considered at all times by the tester anyway, but having them specifically presented would make things clearer and leaves less to individual interpretation. The Problem Frames method also highlights considerations for the tester but they felt that these were less evident.

Conclusion

The Problem Frame diagrams were chosen as being easier to interpret than the AND/OR graphs, although the reviewers did not allocate a difficult rating to either on the

questionnaire. Either diagram would therefore be acceptable for using to trace requirements and prepare test plans. The terminology used in KAOS was preferred but all agreed that the frame concerns would probably highlight this information as well but would be harder for them to find.

I believe that focusing only on the excerpt made this review meeting harder to carry out and to fully decide on a preference the testers would need to spend a longer review period with a full functional specification. However, the review sufficiently highlighted the aspects that the testers prefer and require in order to make their tasks easier.

5.3 Proposed Recommendations and Guidelines

My aim is to propose recommendations or guidelines to advise when each method, or how parts of each method, might be utilised to provide the best approach within financial environments. A major consideration for this task was deciding whether to stay within one method or to try to pick out the best sections of each and try to interweave them. A consideration of mixing steps is that each might loose the context that made it a good technique in the first place. Each step within the proposed guidelines had to work with and complement the step either side of it and contribute to a best practice method.

The two methods are based on different concepts at initial glance: the Problem Frames method avoids hierarchical analysis and approaches the problem domain from the viewpoint of projections; the KAOS method is based on a hierarchical approach. However, the goal reduction into a hierarchical model is partitioned and each AND/OR graph after the initial high-level one is a view of the sub-problem domain. Therefore each method decomposes the problem into sub-problems to be solved individually and then the solutions are carefully joined together to create the final solution to the entire problem. Taking this concept further,

if the individual solutions prescribed to a kind of protocol then they can always be reassembled together, regardless of the analysis path that each had individually taken.

Specific financial environments would need to build up their experience to be able to make recommendations to use Problem Frames for scenarios where experience showed that these worked well, and to use the goal-based ideas from KAOS for scenarios where this worked well etc. These would only be recommendations as each person might have set preferences, skills and experiences. As each method is mastered, as Jackson (2001) argues, the analyst will be able to apply the best method through choice of situation and not just because the guidelines recommend it. Jackson acknowledges that his Problem Frame approach does not suit every kind of problem in software development, and in fact he encourages his audience to not look for a single method to fit all problem areas e.g. use a set display screen solution if available in your environment.

The proposed guidelines need to lead towards best practice for the task of Requirements Analysis so concluding from the literature review in chapter 2, they need to meet the following requirements:

- Do not build a solution before the problem is fully known.
- Produce complete and correct requirements.
- Encourage communication between the customer and the developer to help tackle complexity.
- Help the developer to decide on an approach, method or technique.

With this structure in mind the proposed recommendations and guidelines were drawn up, see table 5.1 for full details. The principles of Pressman (1994) and Davis (1993) were still found to be valid and be good advice and so have been incorporated into the proposed guidelines.

My guidelines build on top of their proved knowledge to specifically focus on financial environments and the considerations that they require. The justification for each step follows.

1	Focus only on the problem domain to elicit the requirements. Do not focus on the solution or start to build a solution until the problem is known. Recommendations are: • read available documentation on the problem domain; read any documents that the user has prepared describing their problem / requirement; observe the problem area and users; arrange meetings with all of the stakeholders. • questions to ask are: what; how; why; who; when. • the problem domain should be considered from an agent, a time and a case driven perspective so that all aspects are reviewed.
2	Create a high level model showing the boundary of the problem domain and all of its interfaces. Recommendations are a choice of: • problem diagram (more sophisticated notation than a context diagram so it can represent more information) • AND/OR graph showing the established goals. • context diagram

3	Decompose the problem into manageable parts and create sub-problem models. The decomposition can be hierarchical or made up of overlapping projections, the analysts' experience and preference might dictate this choice. The objective is that each part is not too complex to continue with. Recommendations are: • Projection into problem frames to be used when the sub-problems look to fit clearly into a basic problem frame: commanded behaviour; information display; required behaviour; simple workpieces; transformation. • AND/OR graph break down into goals to be used if the identified sub-problem does not fit clearly into a problem frame or the sub-problem is preferred to be defined in terms of system and private goals.
4	Use your financial environment's catalogued / classified past experience to see whether any requirement specifications can be reused.
5	Arrange meetings with all stakeholder groups to discuss the models and the break up into sub-problem areas. Use this opportunity to elicit information on any dependencies between the sub-problems.
6	If you are unfamiliar or do not have the required skill set required to analyse either specific functionality or technology then ensure that contact is made with a specialist or training is undertaken.

7	Gradually increase the level of detail until all low level detail is known. Recommendations are: • Problem Frames with full frame concern analysis. • Goal based sub-problems analysed according to the KAOS method. The possible ways of handling identified obstacles should also be provided Information on dependencies between sub-problems should be clearly highlighted. Regardless of the method chosen, each sub-problem should be reviewed for the following general considerations: • Overrun, the ability to respond to each external event • Initialisation, the initial state that the machine has to cope with • Reliability, the failure to fit a baseline requirement description • Identities, to ensure distinct identities can be identified where multiple instances occur and are not related to each other • Completeness, do the problem requirement, problem domain and machine description provide the full picture without ambiguities • Obstacles that might prevent the sub-problem being solved • Constraints that need to be imposed on the sub-problem

8	Arrange meetings with all relevant stakeholder groups to discuss and agree the low level detail of the sub-problems. Take this opportunity to review whether any new regulations or functionality are imminent so that the adaptability of the requirements can be considered.
9	Create the final Requirements Specification document putting all individual sub-problems together so that a final solution can be created. All considerations from guideline 7 should be easy to see in the final specification document so that less is left to subjective interpretation. The final solution should now be considered so that all relevant information is provided in the specification. Where domains are in more than one sub-problem their separately defined characteristics should be brought together to ensure consistency.
10	Deposit your experience into your financial environment's catalogued and classified storage system for use in the future. A problem frames catalogue should be kept sectioned by problem frame. A goals register should be kept according to the classification system chosen. Possible choices are by target condition desired or by subject matter.

Table 5.1: The Proposed Guidelines and Recommendations for Financial Environments

Step 1 - In chapter 1 it is highlighted from Robertson and Robertson (1999) that all too often software developers begin to focus on the solution rather than fully understanding the problem. This is also highlighted by Pressman (1994) who attributes the lack of knowledge of

the problem to being a cause of customer dissatisfaction. Jackson (1994) also reiterates the importance of focusing on the problem domain and not the solution.

Step 2 – In chapter 2 it is noted that Pressman (1994) stressed the importance of modelling the information, function and behaviour in his guidelines.

Step 3 – In chapter 2 it is referenced that Pressman (1994) stated that the problem domain should be broken down into manageable chunks in his guidelines. Also, both Problem Frames and KAOS take this approach and it has been shown to help to handle complexity, a financial environment characteristic.

Step 4 - In chapter 2 a specific characteristic of financial environments is to be able to keep a competitive edge and react to market requirements. By having a catalogued and classified storage system the technique of reuse of requirements specifications will help to accomplish this.

Step 5 – In chapter 2 a specific characteristic of financial environments is shown to be complexity. Involving the user stakeholders at a review process is to try to accommodate and handle complexity.

Step 6 - In chapter 2 Davis' (1993) guidelines stress the importance of utilising the skills of the analyst and developer to ensure that the most appropriate approach is taken. This step acknowledges that training or specialist help should be considered before progressing with the full analysis step. Also in chapter 2, a specific characteristic of financial systems is the presence of legacy systems and the reliance on older skill sets. This step acknowledges this characteristic and tries to accommodate it.

Step 7 - In chapter 2 it is noted that Pressman (1994) states in his guidelines that information should progress from high level to low level detail as the process evolves. Both the Problem Frame and KAOS methods specify for the analysis to take place in this way also.

Step 8 – As for step 5, in chapter 2 a specific characteristic of financial environments is shown to be complexity. Involving the user stakeholders at a review process is to try to accommodate and handle complexity. Another characteristic highlighted in chapter 2 is the ability to adapt to changing requirements due to regulations and laws. This review step is designed to incorporate this consideration into the procedure.

Step 9 – In chapter 2 the example of the V2 rocket compared with the Saturn-B moon rocket highlight very well the importance and need for this step.

Step 10 – As for step 4, in chapter 2 a specific characteristic of financial environments is to be able to keep a competitive edge and react to market requirements. By having a catalogued and classified storage system the technique of reuse of requirements specifications will help to accomplish this. A further characteristic of financial systems is the presence of legacy systems where a major obstacle is the lack of documentation. By keeping a catalogue up to date this will try to avoid the problem being repeated in the future.

5.4 Evaluation and Review of Proposed Guidelines

A follow up review meeting with the designers / analysts was held with a focus on the guidelines and recommendations proposed in table 5.1. The discussion followed the steps through and there was an agreement that the progression of tasks was logical. There was agreement that these guidelines were useable and would help to improve the correctness and consistency of requirements specifications. The focus on discussing the requirements with the stakeholders at two separate stages of the process was agreed as a good guideline to follow and they also wanted to introduce a third review after step 9, when the full requirements specification had been completed. I agreed with this but instead of adding another step I have expanded step 9 to state that the first draft document would be produced and should then be

reviewed and receive comments, just as any other document in the development lifecycle should. The final recommendations and guidelines can be seen in appendix I.

This review meeting was a good way of gaining initial feedback on the guidelines and proposals but the best review would be for them to be used in a financial environment for a period of six months or so. This would allow the recommendations to be made more specific and to be expanded upon. The guidelines and recommendations should be considered as a working document and periodic review and refinement should be made to strengthen them.

Chapter 6 Summary and Conclusions

6.1 Summary of the Research Project

Drawing together the conclusions highlighted throughout this project from the literature review, the case study and the case study review meetings it has been shown that there is not one single method that will suit all people for all situations when carrying out Requirements Analysis in a financial environment.

The summaries for chapters 2 and 4 show the advantages and disadvantages of the two methods, Problem Frames and KAOS. The discussion in chapter 5 summarises the conclusions drawn regarding guidelines and procedures for Requirements Analysis in a financial environment that take into account all issues discussed throughout the project.

The basis for this project was financial environments and whether they require special considerations when drawing up procedures and guidelines. I believe that this project has shown that they do need special consideration. The guidelines stress that communication with the stakeholder is very important to ensure that the complexity of the requirements is handled correctly. This requirement is not so important in say, gathering the requirements for a lift system where the stakeholder will have quite simplistic requirements. Further, still comparing with the more mechanical environments, these would not have to frequently change to keep a competitive edge. Users rarely insist that their lift systems have the latest features and once installed they will stay in place for a long time. Financial environments also have constantly changing regulations to comply with, at the very least on a yearly basis. This is not the case with a post room system or security alarm system where regulations are rarely introduced from decade to decade.

As this project has shown, the specific characteristics of financial environments mean that they do require specially considered procedures and guidelines.

6.2 Guidelines and Recommendations for Financial Environments

After being reviewed, the proposed guidelines and recommendations for the task of Requirements Analysis in a financial environment can be found in appendix I. It is hoped that these will be utilised within my financial environment and in fact be of use in any financial environment to enrich the Requirements Analysis task and ultimately improve the quality of software. As stated in section 5.4, these guidelines should be seen as a working document and periodically be reviewed and enhanced to keep abreast of new methods and techniques that have proved successful for Requirements Analysis.

In order to be able to use the recommended methods of Problem Frames and KAOS, full training in each method would have to be undertaken as the analyst would require pre-requisite knowledge before being able to apply the methods successfully. A further consideration is that the financial environment about to start using these recommendations and guidelines would need to first consider their cataloguing and classification system so that they can fully gain from reusing previous requirements specifications.

6.3 Suggested Future Work

Identified limitations of this research project have been highlighted throughout the work and the addressing of these could form the basis of future work. Therefore, a selection of future research that I would like to suggest, to further enhance Requirements Analysis within financial environments, consists of the following.

Experience of the Reviewers: The basis of this research was that the reviewers, and myself, had experience of working in financial environments. However, there was no previous knowledge of the two methods under review, Problem Frames and KAOS. A suggestion for future work is that this research is progressed by people with experience in the methods, but not necessarily of financial environments. This would then help to address some of the limitations that were experienced.

Using the Guidelines and Recommendations: The guidelines and recommendations have not been followed in a financial environment for Requirements Analysis as this project did not extend to this. One of the limitations highlighted was that only an excerpt of the requirements documentation could be reviewed. Further developments and observations could be drawn from piloting the guidelines and recommendations that would then allow enhancements to be made.

Testing the Methods for Adaptability to Changing Requirements: One of the specific characteristics of financial environments is the requirement to comply with regulations and laws, which requires the ability to adapt to changing requirements at short notice. A highlighted limitation of this project is that there was not enough time to assess the two methods and their ability to handle this characteristic.

Aframes: Architectural frames as mentioned in section 2.3. Rapanotti et al. (2004) argued that the Problem Frame method had some unaddressed disadvantages that would stop it being widely used in commercial, mainstream software development. One of these was that in only focusing on the problem domain and not acknowledging the solution domain, current commercial practice of working with existing components, structures and architectural styles is alienated. Rapanotti et al., (2004, p82) state that the 'intention of AFrames is to provide a practical tool for sub-problem decomposition' that would address this disadvantage. The concept brings together problem frames and architectural style. It therefore needs to be

investigated to decide whether the use of Aframes can enhance software development within financial environments.

Composite Frames: As mentioned in section 2.3, Composition Frames are used to overcome inconsistent requirements by addressing them at the analysis stage rather than allowing an inconsistent set of solutions to be assembled. This area can be researched further and applied to requirements emanating from different problem frames leading on from the examples in Laney et al., (2004).

GRAIL: As mentioned in section 2.4, to support the use of KAOS in the real world, a tool called GRAIL was developed to provide an environment which displays views of the specification in the form of graphics, text, abstract syntax and an object base view as described in Darimont et al. (1997). This project did not explore the use of this tool but further work could investigate its use and document the gains that it produces when applying this method.

Goal reduction approaches: The description's of the KAOS method describe the three goal reduction approaches of (i) agent-driven, to reduce into agent prescribed goals; (ii) case-driven, to reduce into specific cases ranging from normal to exceptional; (iii) time-driven, to reduce into successive goals that should happen in order. In this project I found myself often considering the agent-driven reduction approach. However, further research to determine whether an alternative approach would have made an improvement to the requirements specification could be carried out.

References

Anton, A.I. (1996) 'Goal-Based Requirements Analysis' *Requirements Engineering, Proceedings of the second InternationalConference on*, pp 136 – 144.

Anton, A. I. and Potts, C. (1998) 'The Use of Goals to Surface Requirements for Evolving Systems', *Software Engineering*, *Proceedings of the 20th International Conference on,* pp 157 – 166.

Ashry, N.Y. and Taylor, W.A. (2000) 'Requirements Analysis as Innovation Diffusion: A Proposed Requirements Analysis Strategy for the Development of an Integrated Hospital Information Support System', *System Sciences, Proceedings of the 33rd Hawaii International Conference on,* pp 1 – 10.

Bell, T. E. and Thayer, T.A. (1976) 'Software Requirements: Are They Really a Problem?', *Software Engineering*, *Proceedings of the 2nd International Conference on,* pp 61 – 68.

Bleistein, S. J., Cox, K. and Verner, J. (2004) 'Requirements Engineering for e-business Systems: Integrating Jackson Problem Diagrams with Goal Modeling and BPM', *Software Engineering Conference, Proceedings of the 11th Asia-Pacific*, pp 410 – 417.

Bolchini, D. and Mylopoulos, J. (2003) 'From Task-Oriented to Goal-Oriented Web Requirements Analysis', *Web Information Systems Engineering, Proceedings of the Fourth International Conference on,* pp 166 – 175.

Chen, T.Y., Pak-Lok Poon, Sau-Fun Tang, Tse, T.H., Yu, Y.T. (2002) 'Towards a Problem-Driven Approach to Perspective-Based Reading', *High Assurance Systems Engineering, Proceedings of the 7th IEEE International Symposium on,* pp 221 – 229.

Computing (2006a) 'Finance firms and compliance', June issue, www.Computing.co.uk/2153967 [Accessed 5th March 2007]

Computing (2006b) 'Barclays beats refresh target', July issue, www.Computing.co.uk/2139112 [Accessed 5th March 2007]

Cox K., Hall J. G. and Rapanotti L. (2004) '1st International Workshop on Advances and Applications of Problem Frames' *Software Engineering, Proceedings of the 26th International Conference on*, pp 754 - 755.

Dardenne, A., Fickas, S. and van Lamsweerde, A. (1991) 'Goal-directed Concept Acquisition in Requirements Elicitation', *Software Specification and Design, Proceedings of the Sixth International Workshop on*, pp 14 - 21.

Darimont, R., Delor, E., Massonet, P. and van Lamsweerde, A. (1997) 'GRAIL/KAOS: An Environment for Goal-Driven Requirements analysis, integration and layout', *Requirements Engineering, Proceedings of the Third IEEE International Symposium on*, pp140.

Davis, A. (1993) '*Software Requirements: Objects, Functions, States'*, Hemel Hempstead, Prentice Hall.

Dehong, J. (2001) 'A Method to Reengineer Banking Business By using Information Technology', *Management of Engineering and Technology*, vol 1, p92

Economist (1993) 'On a CREST', *Economist*, vol 328, issue 7818, pp 73 – 74.

Jackson, M. (1994) 'Problems, methods and specialisation', *Software Engineering Journal*, vol. 9, no. 6 (November), pp 249 – 255.

Jackson, M. (1994a) 'Software Development Method', *A Classical Mind: Essays in Honour of C. A. R. Hoare*, Ch 13, A.W. Roscoe ed, Hemel Hempstead, Prentice-Hall.

Jackson, M. (1995) *Software Requirements & Specifications: A Lexicon of Practice, Principles and Prejudices*, Harlow, Addison-Wesley.

Jackson, M. (1995a) 'Problems and Requirements', *Requirements Engineering Symposium, Proceedings of the second IEEE International on*, pp 2 – 8.

Jackson, M. (1997) 'Problem Complexity', *Engineering of Complex Computer Systems, Proceedings of the third IEEE International Conference* on, pp 239 – 248.

Jackson, M. (2001) *Problem Frames: Analyzing and structuring software development problems*, Harlow, Addison-Wesley.

Laney, R., Barroca, L., Jackson, M. and Nuseibeh, B. (2004) 'Composing requirements using problem frames', *Requirements Engineering Conference, Proceedings of the 12th IEEE International*, pp 122 – 131.

Polya, G. A. (1957) *How to solve it* (2nd edition), Princeton, NJ, Princeton University Press.

Pressman, R. S. (1994) *Software Engineering: A Practitioner's approach* (3rd edition), London, McGraw-Hill Publishing Company.

Rapanotti, L., Hall J. G., Jackson, M. and Nuseibeh, B. (2004) 'Architecture-driven Problem Decomposition' *Requirements Engineering Conference, Proceedings of the 12th IEEE International*, pp 80 – 89.

Robertson, S. and Robertson, J. (1999) 'M*astering the Requirements Process'* , Harlow, Addison-Wesley.

The Open University (1997), *M880 Requirements Specification*, The Open University.

Titscher, S., Meyer, M., Wodak, R. and Vetter, E. (2003) *Methods of Text and Discourse Analysis*, London, Sage Publications.

van Lamsweerde, A., Darimon, R. and Massonet, P. (1995) 'Goal-Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt', *Requirements Engineering Symposium, Proceedings of the second IEEE International on*, pp 194 – 203.

van Lamsweerde, A. and Letier, E. (1998) 'Integrating Obstacles in Goal-Driven Requirements Engineering', *Software Engineering*, *Proceedings of the 20th International Conference on,* pp 53 – 62.

van Lamsweerde, A. (2000) 'Requirements Engineering in the Year 00: A Research Perspective', *Software Engineering, Proceedings of the 2000 International Conference on*, pp 5 - 19.

van Lamsweerde, A. (2001) 'Goal-Oriented Requirements Engineering: A Guided Tour', *Requirements Engineering, Proceedings of the fifth IEEE International Symposium on*, pp 249 – 262.

van Lamsweerde, A. (2001a) 'Building Formal Requirements Models for Reliable Software', *Reliable Software technologies*, *Proceedings of the 6th Ade-Europe International Conference Leuven on*, pp 1-20.

van Lamsweerde, A. (2004) 'Goal-Oriented Requirements Engineering: A Roundtrip from Research to Practice', *Requirements Engineering, Proceedings of the 12th IEEE International Conference on*, pp 4 - 7.

Wing, J.M. (1988) 'A Study of 12 Specifications of the Library Problem', *IEEE Software*, vol. 5, issue 4 (July), pp 66 – 76.

Yin, R. K. (1994) *Case Study Research: Design and Methods* (2nd edition), London, Sage Publications.

Index

Appendix A Context Diagram for Export Cash Cover

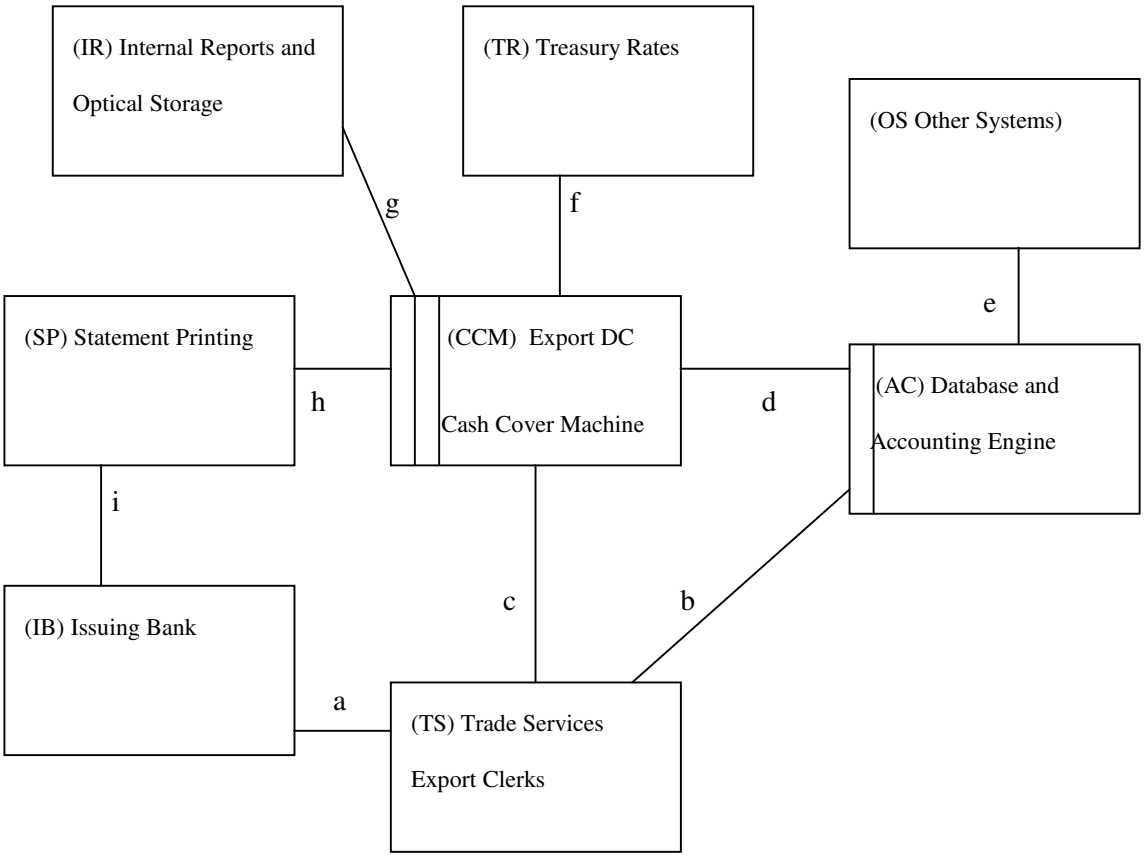
To provide the background for the case study, below is a description of the problem domain and existing system together with the business initial requirements:

Problem domain / existing system

Currently, the ability to record details of cash cover taken against Export Documentary Credits, or Export Documentary Credit Bills does not exist. Accounting entries would have to be made manually and there is consequently no Cash Cover statement generated by the existing Import Export system

Without a solution to this problem, the users would have to manually pass accounting entries, record details of cash cover held manually, and prepare manual statements. This means that there is the possibility of cash cover not being drawn down correctly, and for inaccurate or missing statements due to manual error. It would also be very time consuming to prepare manual statements, and maintain manual records.

Below is the context diagram created as part of the Problem Frame modelling step.



a: {ReqNewExportDC, ReqAmendExportDC, PayExportDCBill}

b: {CreateExportDC, AmendExportDC, DrawDownExportDCBill}

c: {RecordCC, AmendCC, EnquireCC, DrawDownCC)

d: {UpdateDB,CreateAccountingEntries, QueryDB}

e: {ReqRspToOtherSys} OS!{ReqRspToAC}

f: {UpdateCurRate}

g: { SendReportData, FormatReports, StoreReports}

h: {SendStatementDetails}

i: {FormatCustomerStatements, PostCustomerStatements}

Notes to accompany the context diagram:

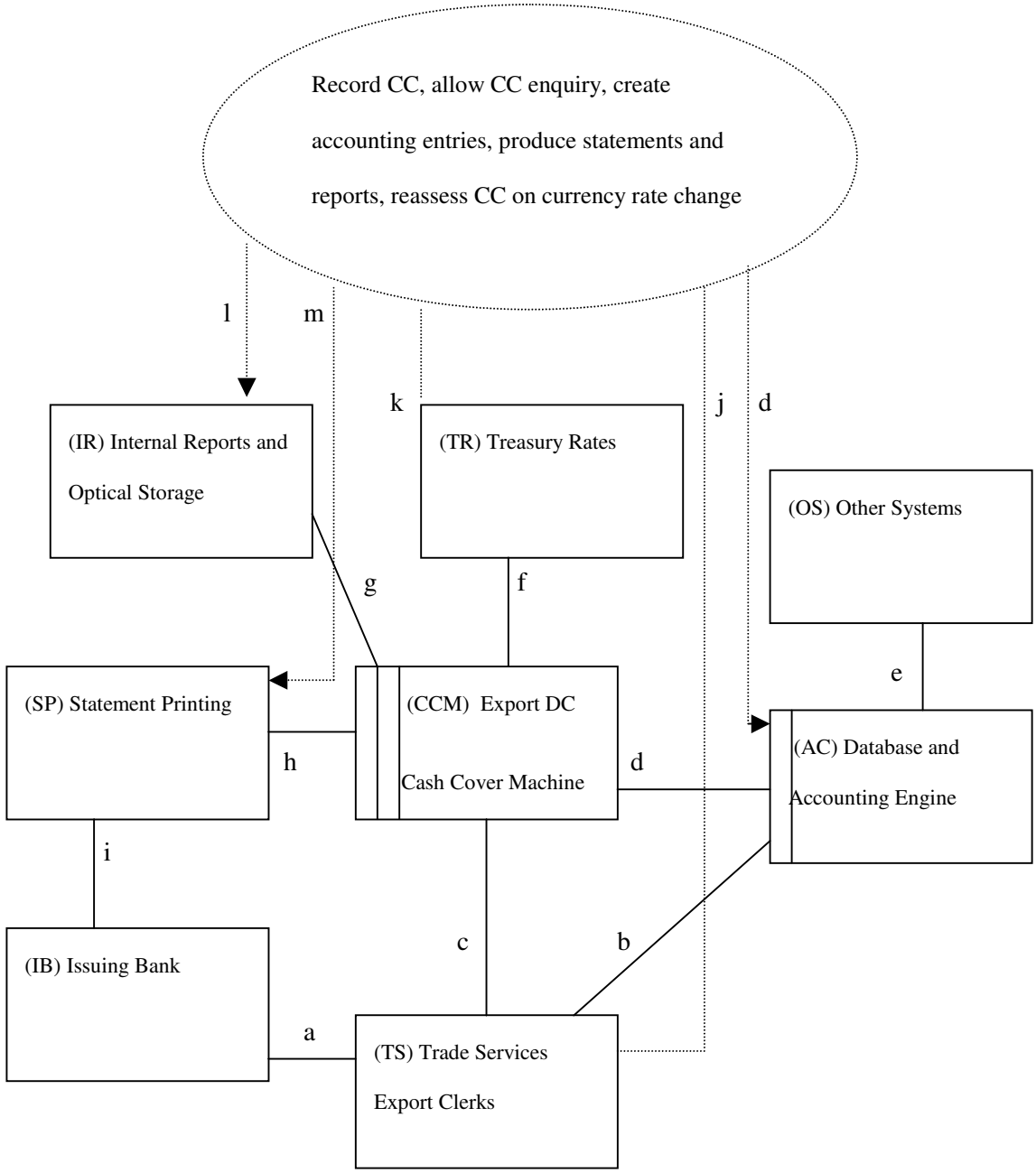
1) For the activity of the (TS) Trade Services Export Clerks I have not included the general updating and enquiry of the existing database and accounting engine as this functionality already exists as part of the current system.

2) The post office is a connection domain in between SP and IB but this has been omitted as it is not deemed to materially affect the problem domain.

3) Phenomena e has been included to show that the database and accounting engine already exist and have to satisfy existing conditions, however, it will not be included in the subproblem breakdown. It will be included in the review of the final solution though.

Appendix B Problem Diagram for Export Cash Cover

Below is the problem diagram created as part of the Problem Frame modelling step.



a: IB! {ReqNewExportDC, ReqAmendExportDC, ReqExportDCInfo, PayExportDCBill}

b: TS! {CreateExportDC, AmendExportDC, DrawDownExportDCBill}

c: TS! {RecordCC, AmendCC, EnquireCC, DrawDownCC}

d: CCM! {UpdateDB,CreateAccountingEntries, QueryDB}

e: AC! {ReqRspToOtherSys} OS!{ReqRspToAC}

f: TR! {UpdateCurRate}

g: CCM! { FormatReports, SendReportData,} IR! {StoreReports}

h: CCM! { FormatCustomerStatements, SendStatementDetails} SP! {StoreCustomerStatements}

i: SP! {PostCustomerStatements}

j: TS! {SupplyExportDCDetails, EnquireExportDCdetails}

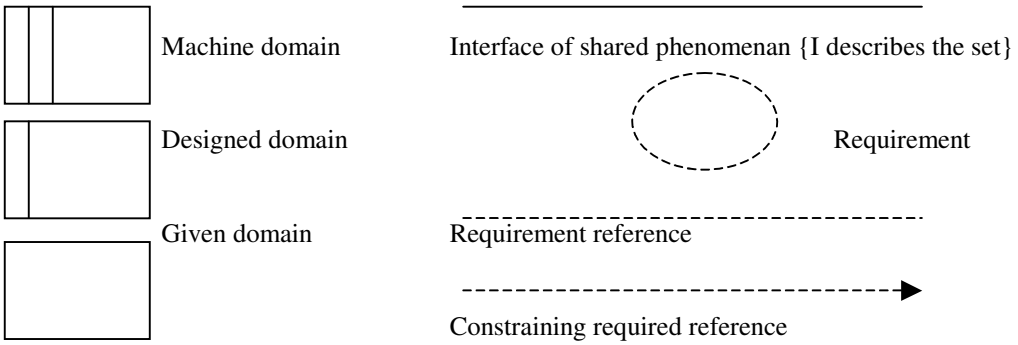
k: CCM! {RespondToCurRateChange}

l: CCM! {SupplyReportData}

m: CCM! {SupplyStatementData}

Key for all problem diagrams:

Domain types: C = causal, B = biddable, X = lexical.

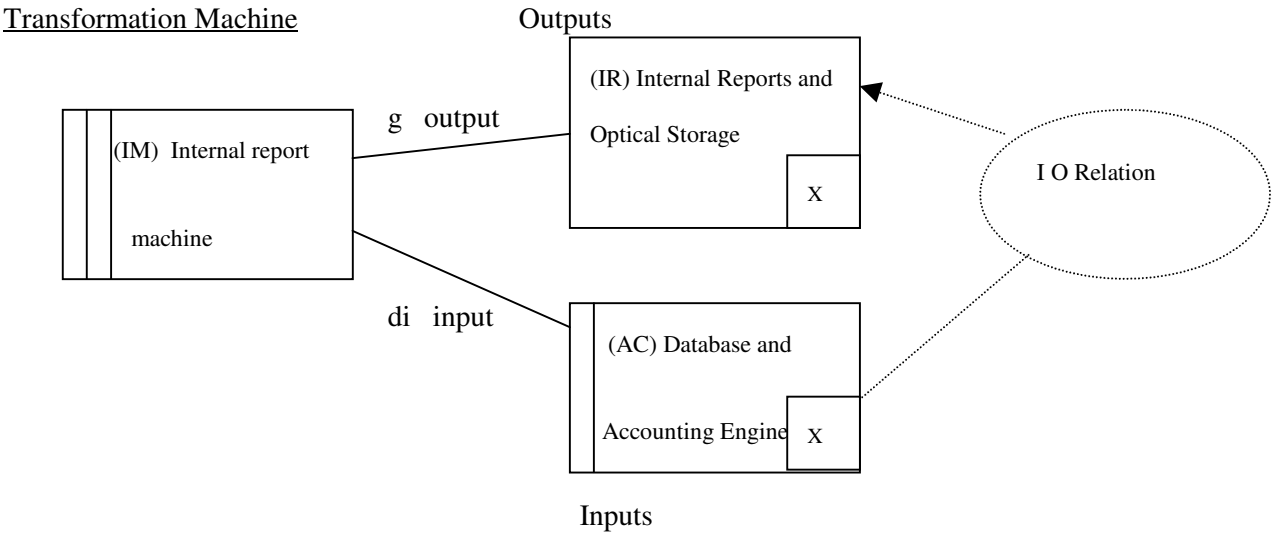


Appendix C Sub-problem Diagrams

Below are the sub-problem diagrams, broken down in to simple problem frame level, created as part of the Problem Frame analysis and modelling steps:

Transformation Problem Frame

Export CC Internal Report sub-problem diagram:



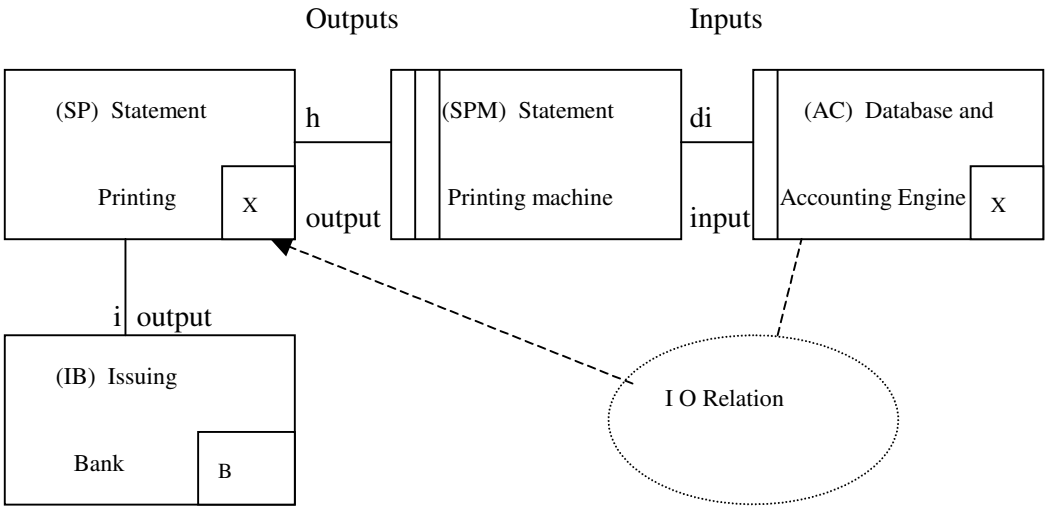
This sub-problem has two lexical domains marked X. This problem frame was chosen as the computer-readable data from AC is required in order to produce the particular format of the cash cover internal reports.

di: IM! {QueryDB} (part of d)

g: IM! {FormatReports, SendReportData} IR! { StoreReports}

Export CC Statement Print sub-problem diagram:

Transformation Machine



This sub-problem has two lexical and one biddable domains marked X and B respectively. This problem frame was chosen as the computer-readable data from AC is required in order to produce the particular format of the customer cash cover statements.

di: SPM! {QueryDB} (part of d)

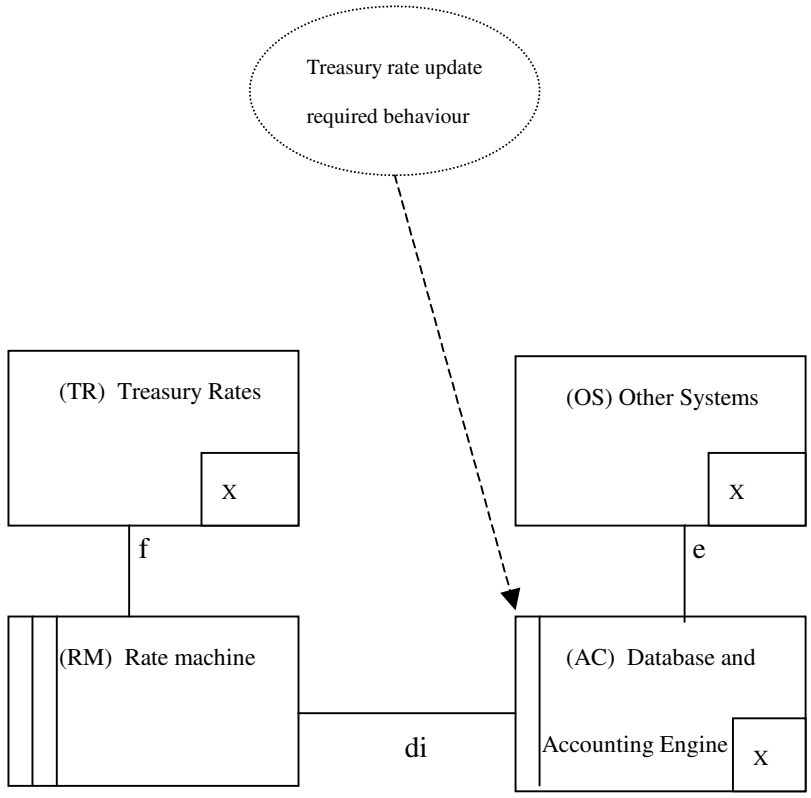
h: SPM! {FormatCustomerStatements, SendStatementDetails} SP! {StoreCustomerStatements}

i: SP! {PostCustomerStatements}

Required Behaviour Problem Frame

Export CC Treasury Rates sub-problem diagram:

Required Behaviour Machine



This sub-problem has three lexical domains marked X. This problem frame was chosen as the exchange rates used by the system are part of the real world whose behaviour has to be controlled to satisfy the financial controls required of the system.

di: RM! {UpdateDB} (part of d)

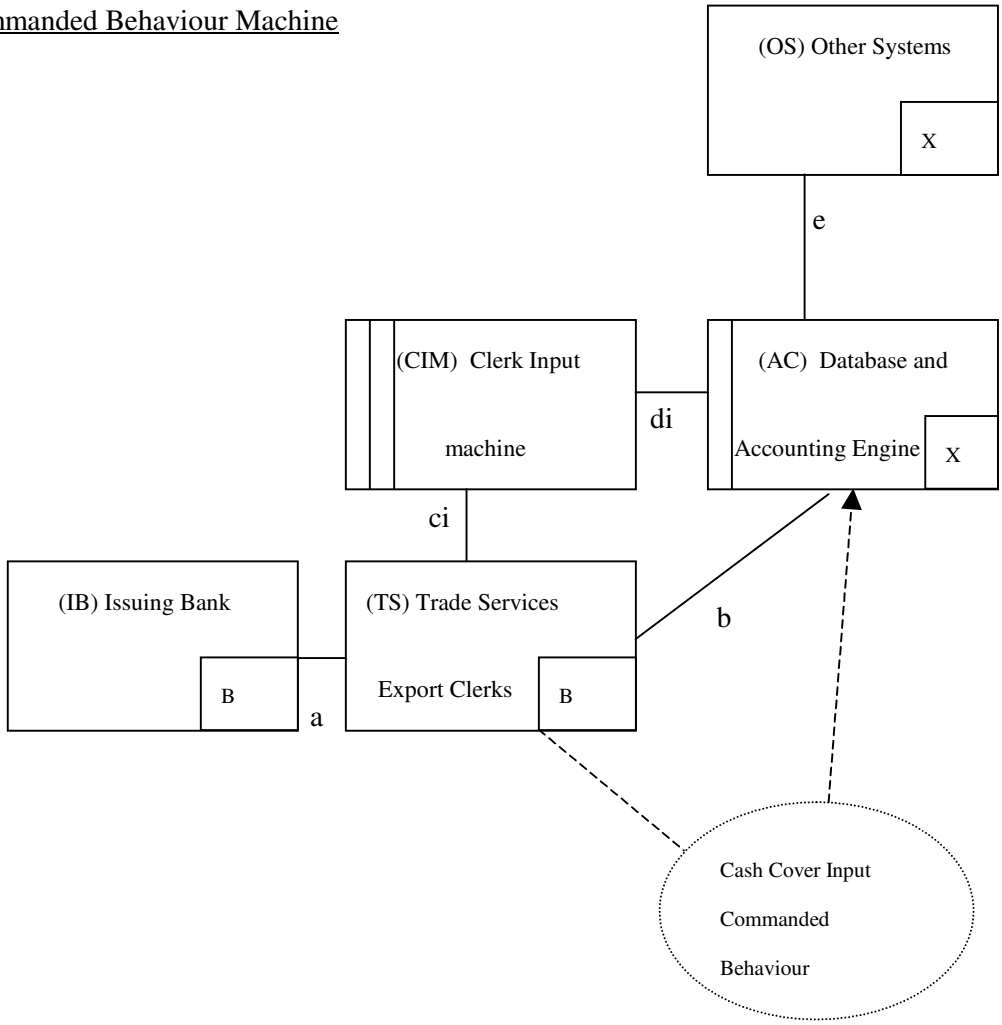
e: AC! {ReqRspToOtherSys} OS!{ReqRspToAC} (to be considered in this sub-problem due to timing that rates might be required for other systems.)

f: TR! {UpdateCurRate}

Commanded Behaviour Problem Frame

Export CC Clerk Input sub-problem diagram:

Commanded Behaviour Machine



This sub-problem has two lexical and two biddable domains marked X and B respectively. This problem frame was chosen as the real world data contained in AC has to be controlled in accordance with the commands issued by TS, who will often be acting on behalf of IB.

a: IB! {ReqNewExportDC, ReqAmendExportDC, ReqExportDCInfo, PayExportDCBill}

b: TS! {CreateExportDC, AmendExportDC, DrawDownExportDCBill}

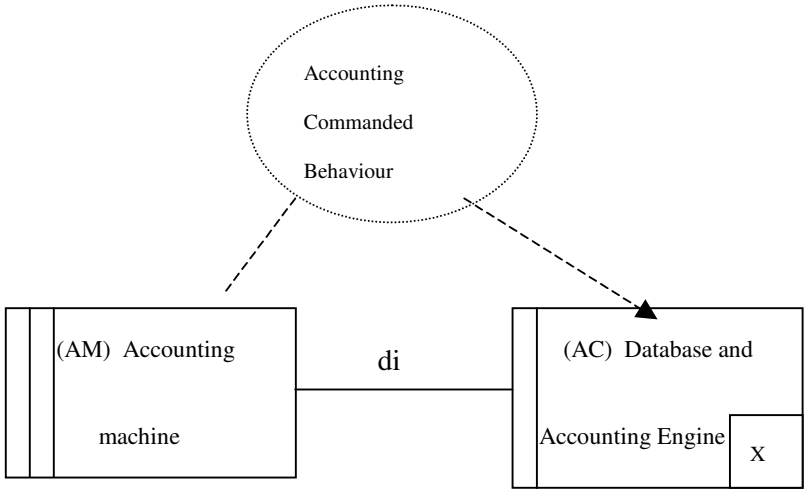
ci: TS! {RecordCC, AmendCC, DrawDownCC} part of c

di: CIM! {UpdateDB}

e: AC! {ReqRspToOtherSys} OS!{ReqRspToAC} (to be considered in this sub-problem due to the way that the new data is stored in consideration of whether it will be required for other systems.)

Export CC Accounting Entries sub-problem diagram:

Commanded Behaviour Machine



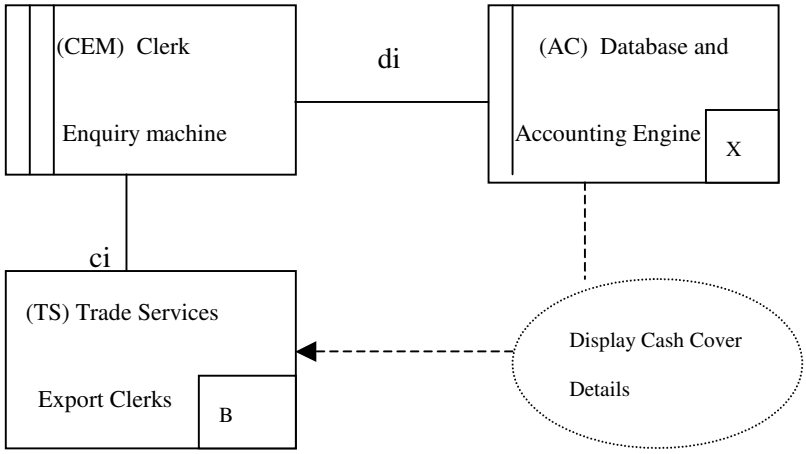
This sub-problem has one lexical domain marked X. This problem frame was chosen as the real world accounting data contained in AC has to be controlled in accordance with the commands issued by AM. In this case the machine will be issuing the commands, this is a feature of financial environments that are built not to require manual intervention.

di: AM! {CreateAccountingEntries}

Information Display Problem Frame

Export CC Clerk Enquiry sub-problem diagram:

Information Display Machine



This sub-problem has one lexical and one biddable domain marked X and B respectively. This problem frame was chosen as information about the real world data contained in AC is continually required by TS.

ci: TS! {EnquireCC} part of c

di: CEM! {QueryDB}

Appendix D Identified Goals for Export Cash Cover

List of goals identified after review with business user

System Goals:

- Have an accessible list of issuing banks for which cash cover is required.
- Record details of CC when taken to enable the issuance of an unconfirmed Export DC or when taken as security for adding confirmation to an Export DC
- Automate accounting entries to action the cash cover recorded.
- Ensure amount of CC held is sufficient considering changing currency exchange rates where cross currency deals are in place.
- Produce statements and reports of the cash cover details and positions.

Private Goals:

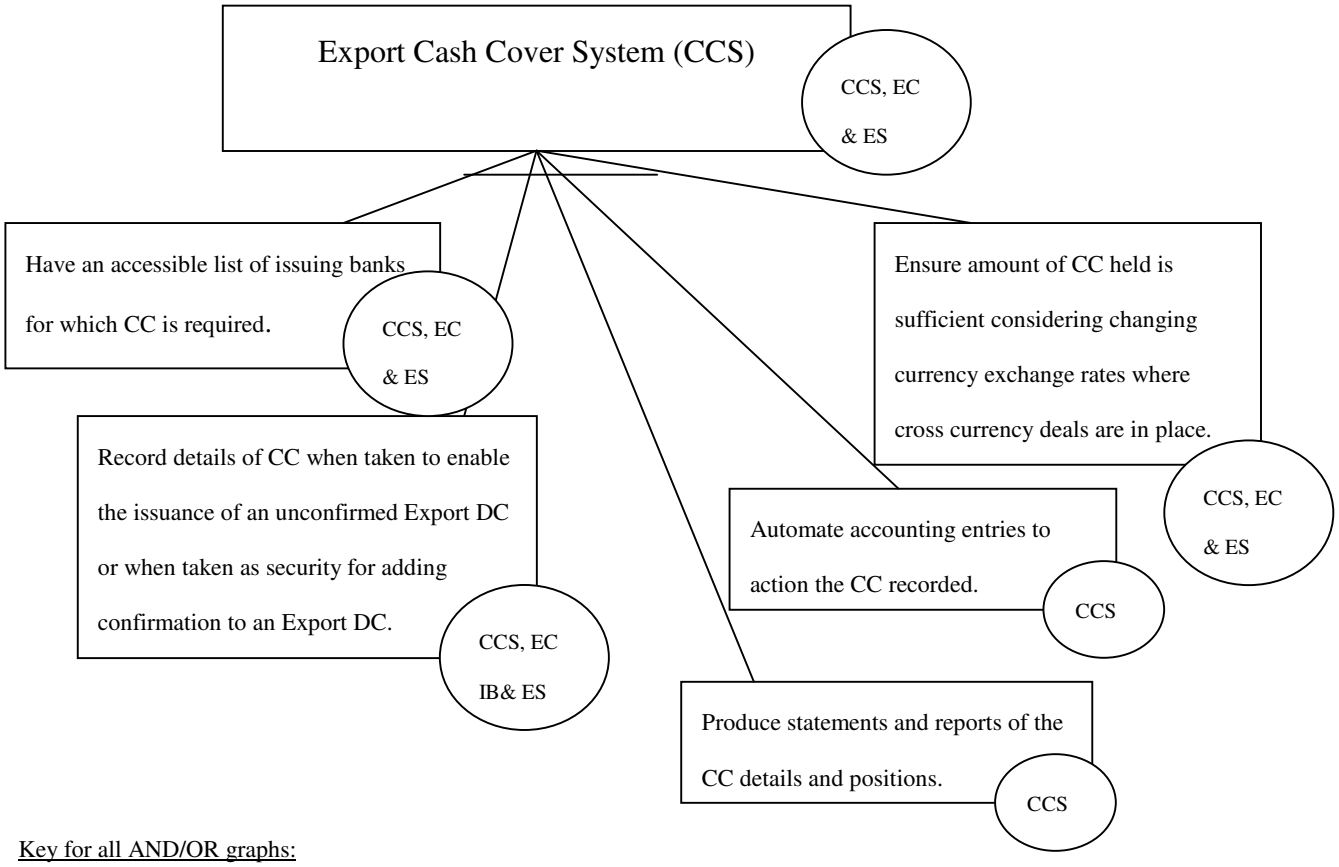
Export clerk and supervisor:

- System is available for use from 07:00 GMT until 19:00 GMT five days a week.

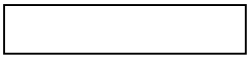
Appendix E AND/OR Graphs for Export Cash Cover

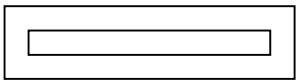
The first diagram is at a high level showing only the initially specified goals in their AND reduction link. Each branch has then been reduced and displayed separately to ensure that the diagrams are not too complex.

High level main AND/OR graph

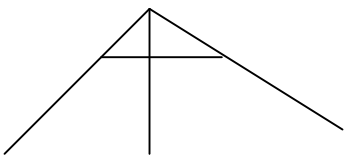


Key for all AND/OR graphs:

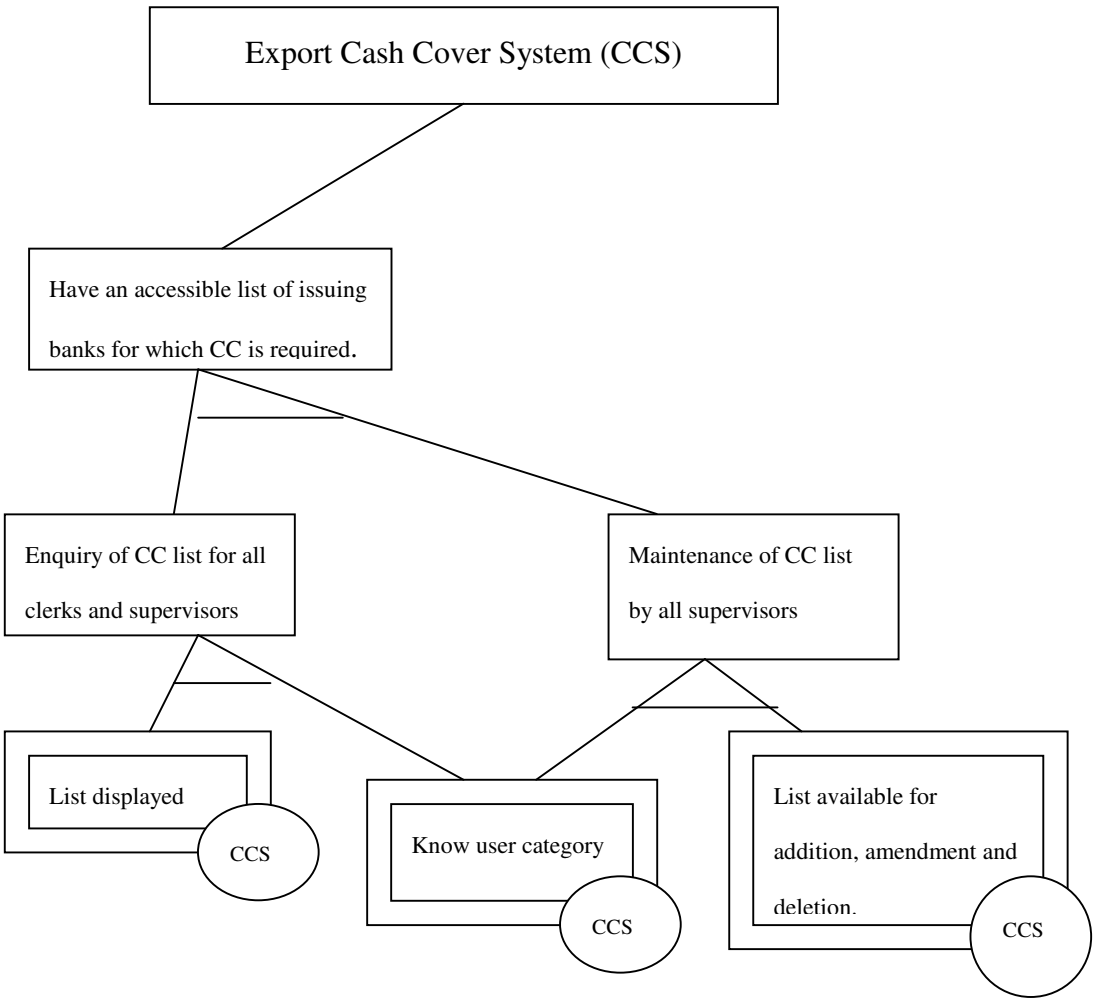
Goal: 

Leaf Goal: 

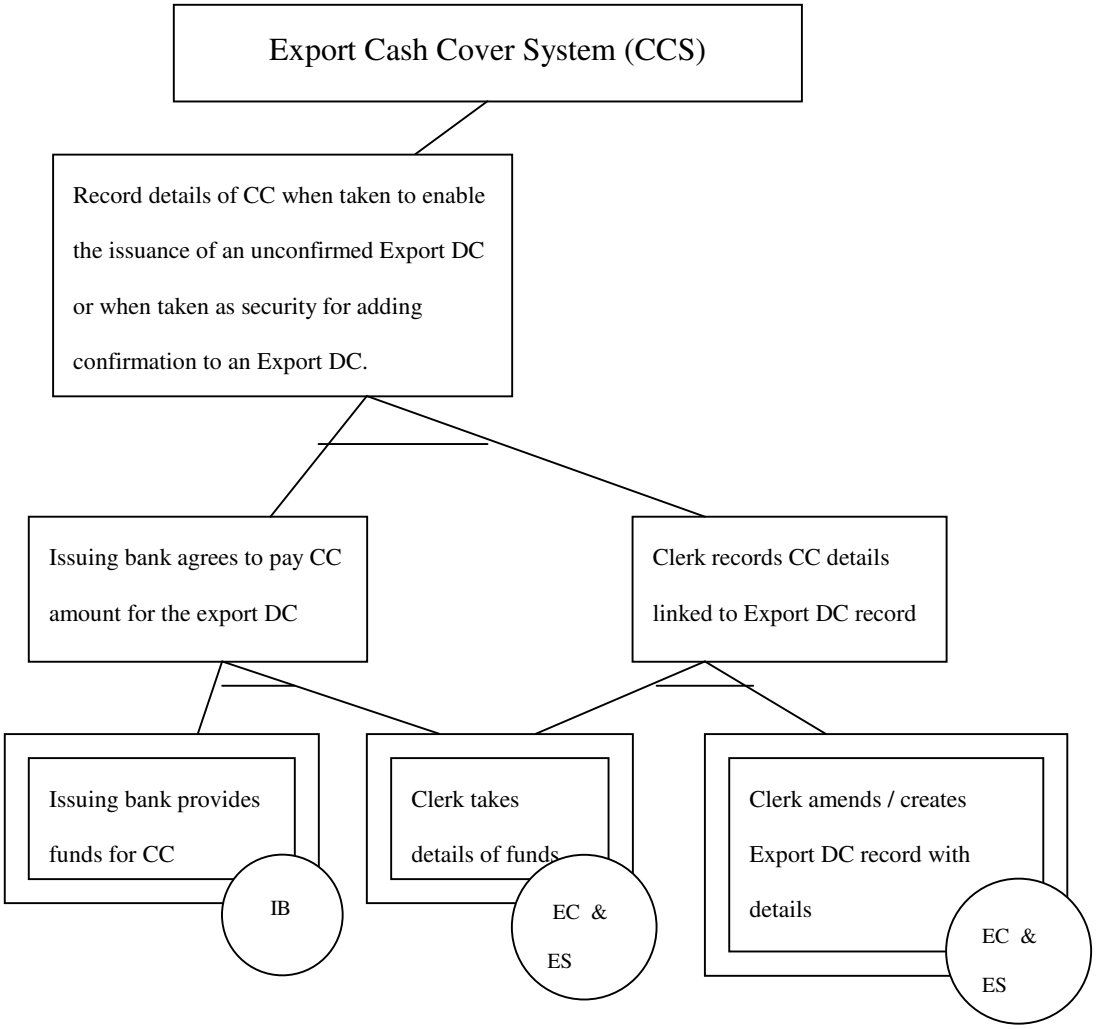
Responsible agent:  (IB: issuing bank; EC: Export clerk; ES: export supervisor; CCS: CC system)

Reduction link: 

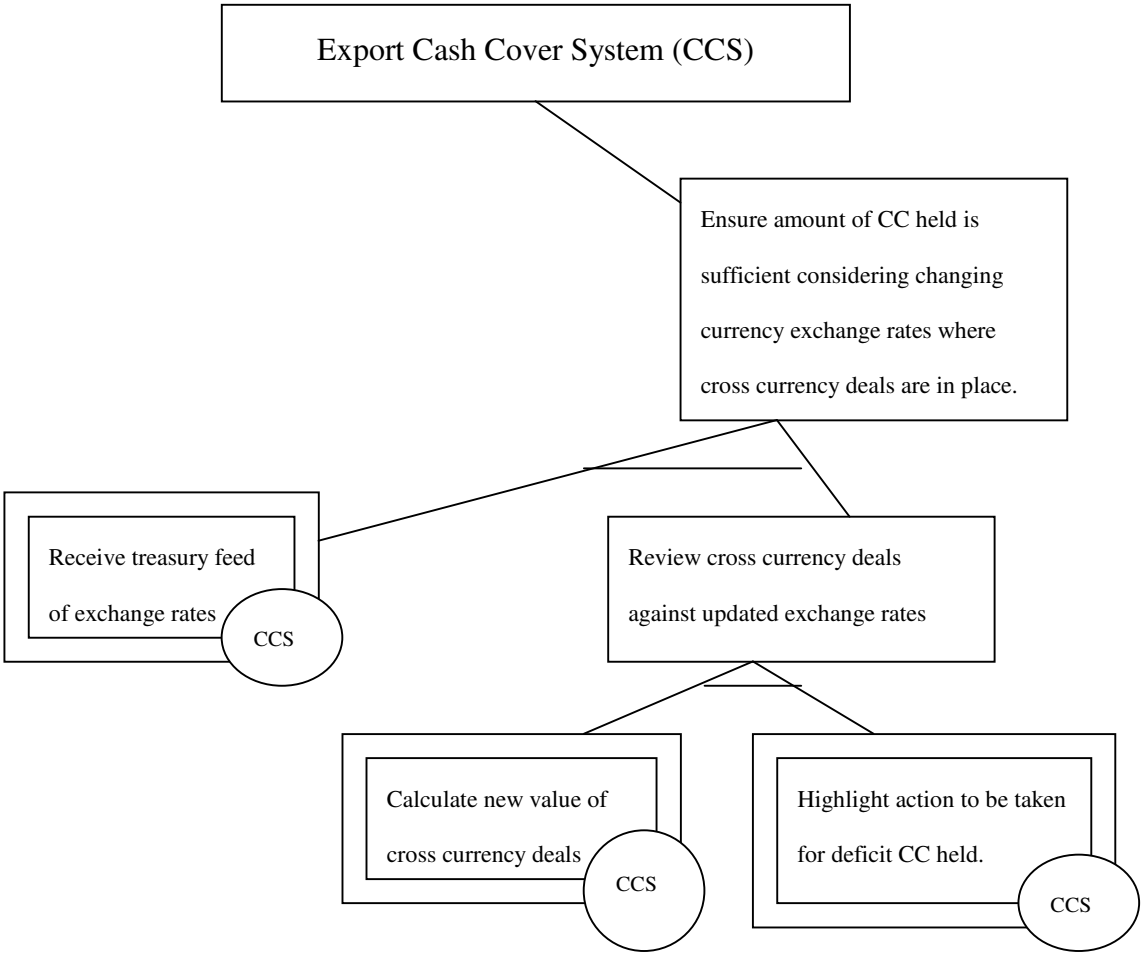
‘Have an accessible list of issuing banks for which CC is required’ sub-graph:



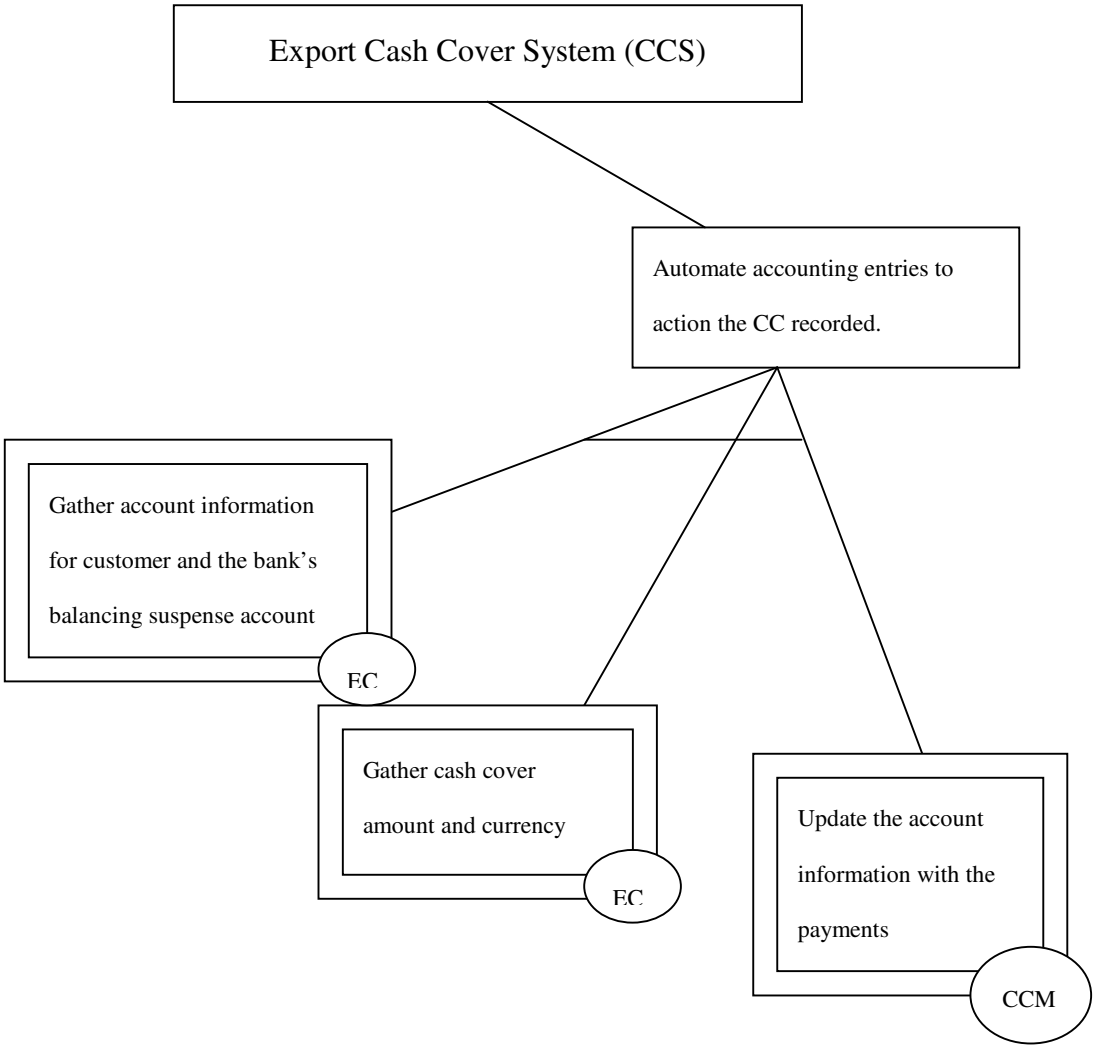
'Record details of CC when taken to enable the issuance of an unconfirmed Export DC or when taken as security for adding confirmation to an Export DC' sub-graph:



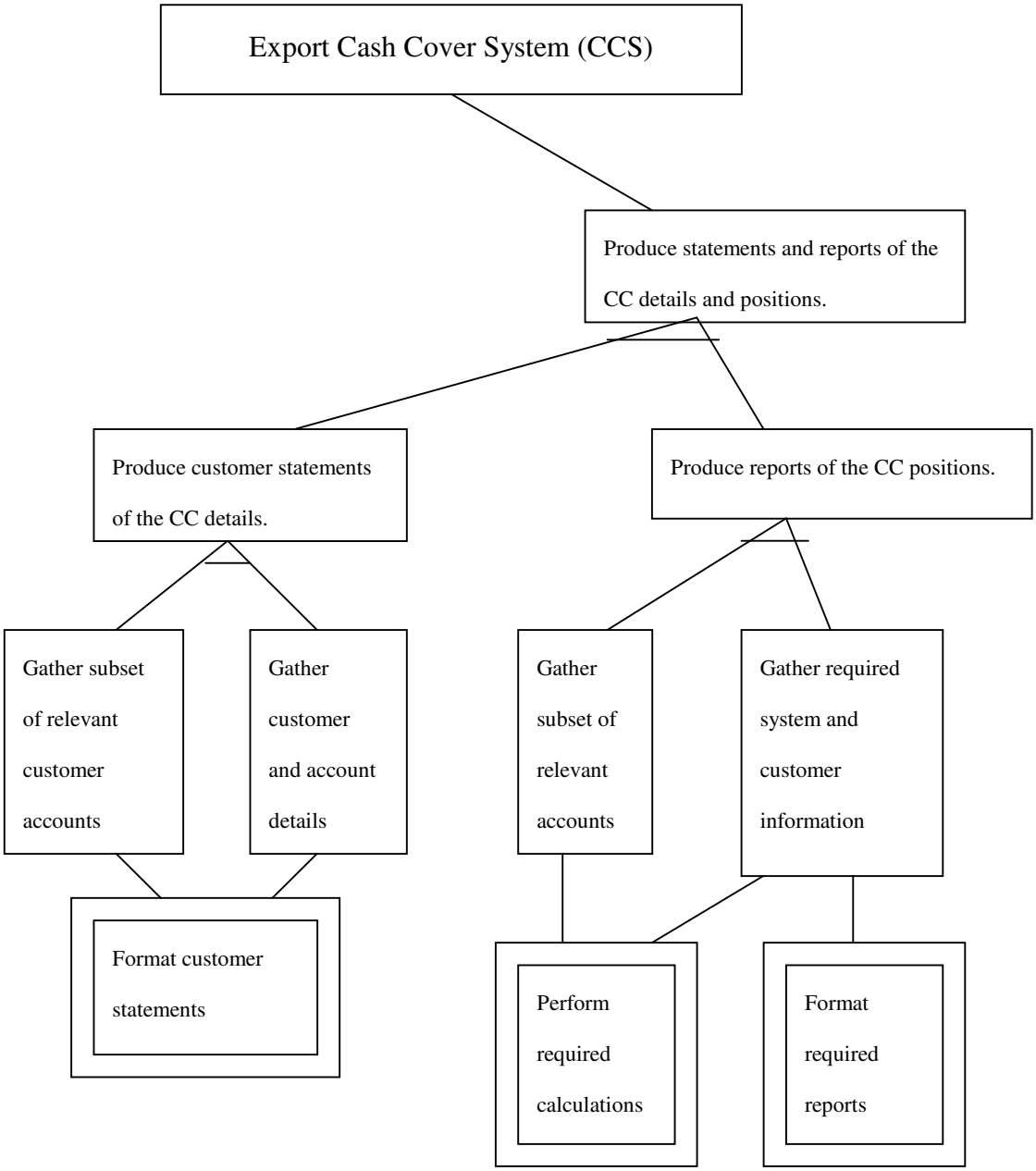
‘Ensure amount of CC held is sufficient considering changing currency exchange rates where cross currency deals are in place’ sub-graph:



‘Automate accounting entries to action the CC recorded’ sub-graph:

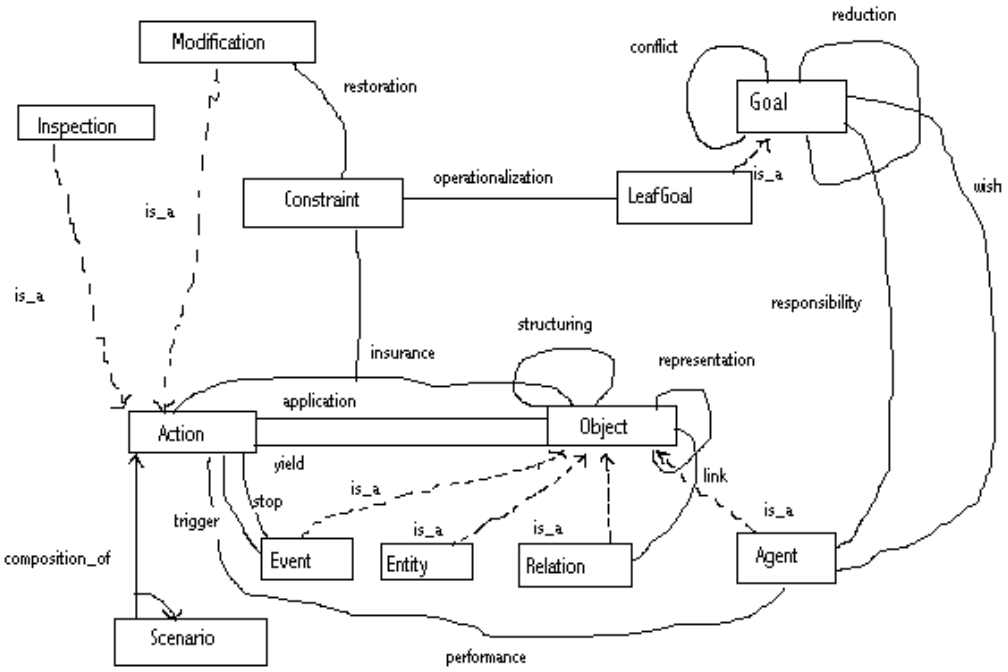


‘Produce statements and reports of the CC details and positions’ sub-graph:



Appendix F The KAOS Meta-Model

Taken from Dardenne et al. (1991)

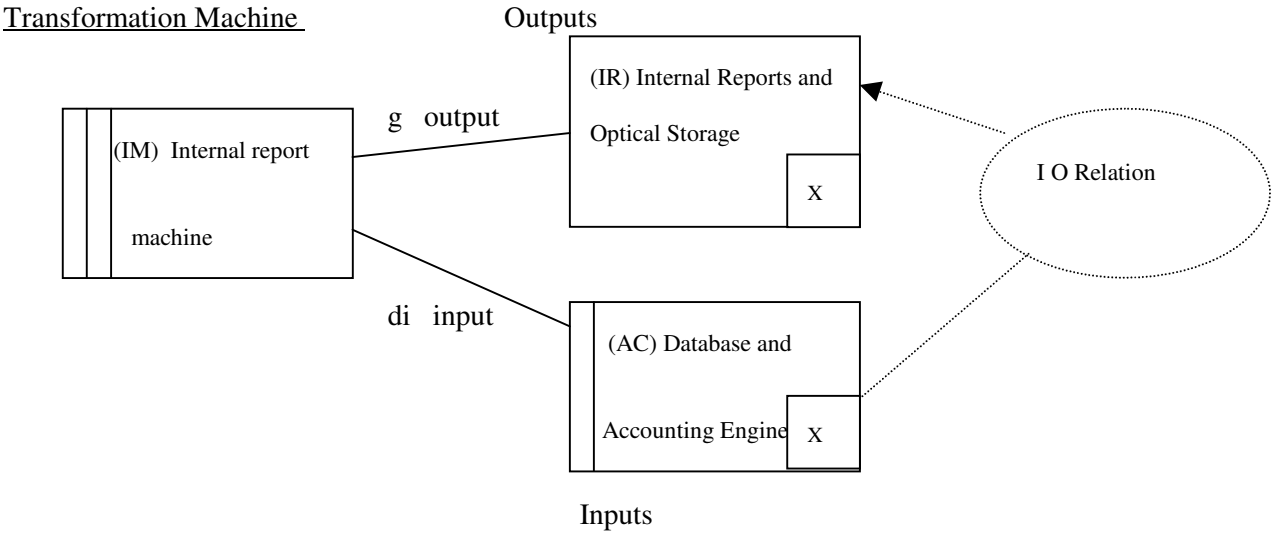


The KAOS conceptual meta-model

Appendix G The Review Documentation

Problem Frames

Export CC Internal Report sub-problem diagram:

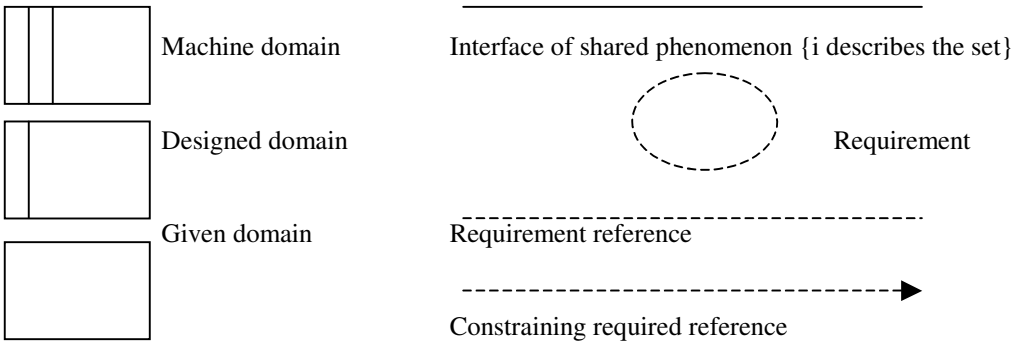


Controlling domains and sets of phenomena:

di: IM! {QueryDB}

g: IM! {FormatReports, SendReportData} IR! {StoreReports}

Key:



The requirement

The requirement is for a report titled 'Cash Cover X-CCY Reconciliation Report' that is produced daily. The report will start at page 1 and a new section with headings must be produced for every customer. Each page should contain identical headings, the format and further detail is shown in the example report below. If a report goes over one page the headings will be the same except for the page number.

The detail on the report must contain information of all outstanding DC records, the DC currency and amount outstanding, the cash cover currency and amount held and the exchange rate for the two currencies involved. For each customer, the cash cover should be totalled at the end of the customer section break as well as the DC amount in the ledger balance. The difference should then be calculated and printed. The next customer records can then begin.

The domain – This description covers two lexical domains, the input domain and the output domain. Lexical domain is a term used to represent the physical data.

The input domain: This is information from the accounting database that is stored across several indexed database files. It consists of customer data and export business information including related documentary credits, bills and cash cover. It must be accessible in a way that for each customer the associated export DCs can be located together with all cash cover and other listed details. It also contains system information like the current date, the working calendar and currency details e.g. number of decimal places for each currency.

The output domain: This contains the report lines ready for printing and storage contained on a spool file with embedded control characters for line and page breaks. An example report is shown below. Each line of data is less than 132 characters and ends with a carriage return and line feed command representation. Each page will contain 60 printable lines.

The specification

The frame concern for the transformation problem frame requires a solution that traverses the input and output domain efficiently, avoiding:

- Multiple visits to the same data.
- Unnecessary visits to irrelevant data.
- Having to save a large amount of data in transit between input and output.

The internal report machine (IM) to solve this transformation sub-problem must have indexed access to the Export DC master information, the Export DC bill information, the new Export DC cash cover information, the customer details, the customer address information and the system information for the current date, the working calendar and currency details e.g. number of decimal places for each currency.

A sorted Export DC master information database table will be the primary input as only customer details with export DC business will require accessing. The different Export DC database tables all contain an index and will be required to be sorted in the same order prior to being accessed as input. This will ensure that no unnecessary reading will happen and the pointer can be set and incremented once only through the file. The customer information database tables are stored by separate references and so cannot be prearranged into a specific order. Multiple visits reading through this file cannot be avoided.

The general concerns that need addressing are:

Initialisation – The total amounts need to be zeroed every time a customer change takes place and totals maintained whilst accessing the same customer. The difference calculation is then made when the end of customer is reached.

Reliability – The Export DC master information database table will be the primary input for the processing. If any mandatory information, like the customer details, the Export bill details or the cash cover details are missing, the machine must handle this by reporting the issue on a separate report and continuing on to the next Export DC master instance.

CASH COVER ACCOUNT EXCEPTION REPORT DDMMMYY HH:MM PAGE NN

Customer Issuing Bank CBID	SHORT NAME	Cash Cover A/C NO	Cash Cover Currency
ABCD	ABC Bank Ltd	nnnnnnnnnn	EUR

DC Number	DC CCY & AMT Outstanding	CC Amount Held As per DC Record	Input Rate As per DC Record
DCXXXXXX	GBP 100000.00	EUR 150000.00	1.5
DCXXXXXY	GBP 100000.00	EUR 150000.00	1.5
DCXXXXXY	EUR 100000.00	EUR 100500.00	1.0

Total CC Account EUR 400000.00 Ledger Balance EUR 400500.00
Difference EUR 500.00

Customer Issuing Bank CBID	SHORT NAME	Cash Cover A/C NO	Cash Cover Currency
EFGH	Bank Ltd	nnnnnnnnnnnn	GBP

DC Number	DC CCY & AMT Outstanding	CC Amount Held As per DC Record	Input Rate As per DC Record
DCXXXXXY	USD 100000.00	GBP 150000.00	1.5

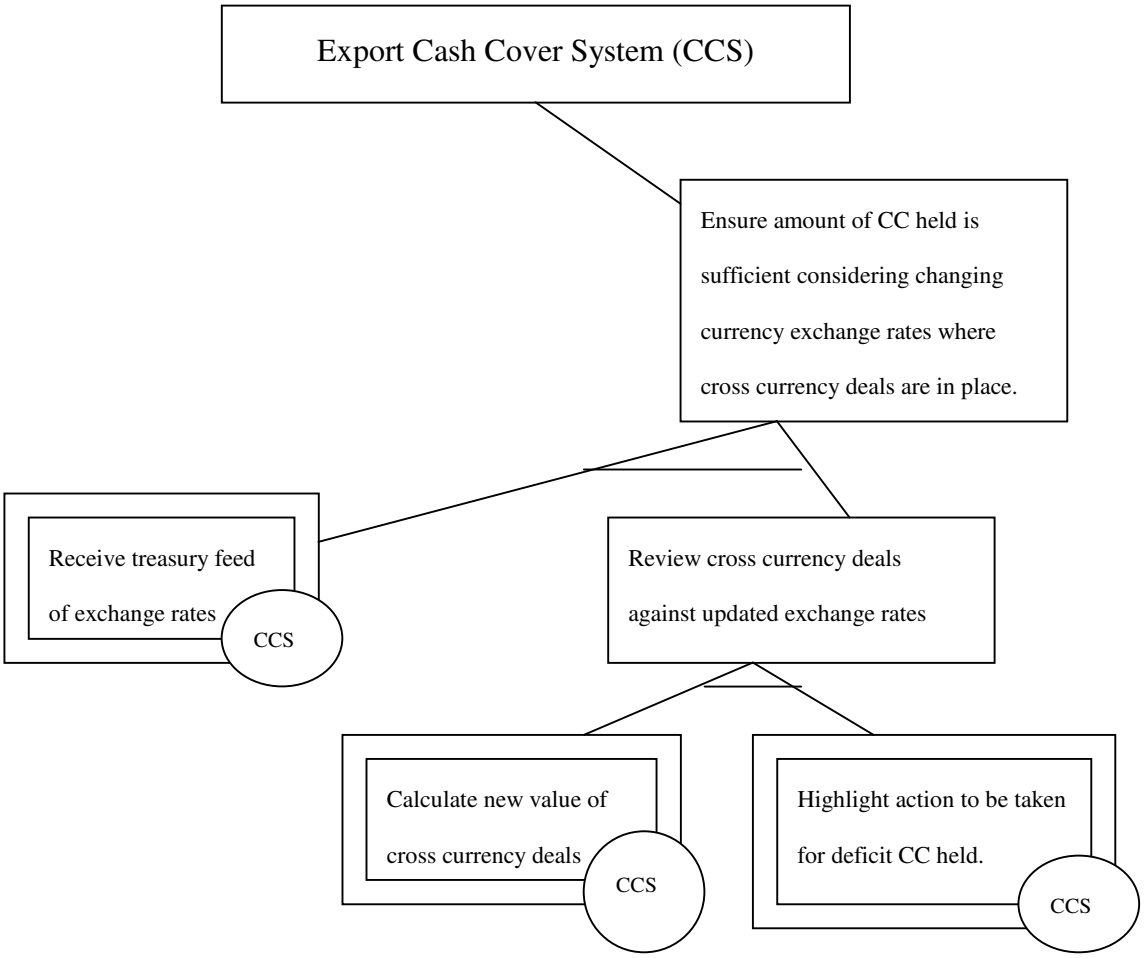
Total CC Account GBP 150000.00 Ledger Balance GBP 200000.00
Difference GBP 50000.00

End of Report

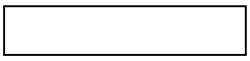
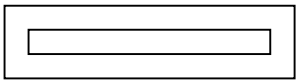
KAOS

System Goal: Calculate new value of cross-currency deals

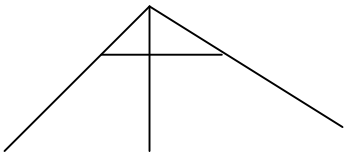
Part of the 'Ensure amount of CC held is sufficient considering changing currency exchange rates where cross currency deals are in place' sub-graph:



Key for all AND/OR graphs:

Goal:  Leaf Goal: 

Responsible agent:  (CCS: Cash Cover system)

Reduction link: 

Type: Achievement goal

Description: When the cash cover currency is held in a different currency to the actual Export Documentary Credit, calculate the value of the cash cover using the updated currency exchange rates to check whether the value is still covered or not.

E.g. If the Export DC is in USD and the related outstanding Export DC bill is still 23,000USD, will the cash cover of 12,460 GBP still cover it? Is the cash cover held too high?

Action: Calculate the value of all instances of cross-currency cash cover held.

Agent: Cash Cover System, CCS

Stakeholders: Financial institution, Accounts department, Export supervisors, Export clerks, Customers.

Obstacles: 1. Updated currency exchange rates are not available.

2. Export DC cash cover or bill information is not available.

Scenarios: 1.1 Transmission method for updating the exchange rates is not available.

1.2 Users responsible for updating the exchange rates fail to do so.

2.1 There is no associated bill information available (database integrity error).

Constraints: None

Preconditions: 1. System goal 'Receive treasury feed of exchange rates' must have completed prior to this action beginning.

2. CCS knows which Export DCs have associated cash cover held in a different currency.

3. Input of export DC cash cover and bills is available.

Postconditions: 1. New value of the cash cover, in the currency of the Export DC, must be available as an input for the system goal 'Highlight action to be taken for deficit CC held' action.

Links: There are goal links to both 'Receive treasury feed of exchange rates' and 'Highlight action to be taken for deficit CC held'.

Specification: The CCS solution will be responsible for carrying out the above action. The obstacles must be addressed by considering the listed scenarios and providing an agreed solution. The solution must be designed to know when the preconditions have been met and to know what action to take if they have not been. Time restrictions should also be considered. The solution must also provide the post-conditions and indicate the consequences

and actions to be taken if this responsibility is not achieved.

The specifications for the two linked goals should be checked against this one to ensure compatibility and consistency.

This specification is to be agreed by all stakeholders. When a solution is in place all stakeholders must also agree the testing procedure.

Appendix H The Evaluation Questionnaire

1. Have you ever had any experience of the Problem Frame method or problem diagrams before?

2. Have you ever had any experience of KAOS or other goal based method before?

3. Which method of requirements elicitation do you use?

For questions 4 and 5, please indicate using a scale of 1 –5 where 1 = easy to interpret and 5 = difficult to interpret.

4. Did you understand the Problem Frame diagram?

5. Did you understand the AND/OR graph of the KAOS method?

For questions 6 and 7, please indicate using a scale of 1 – 5 where 1 = useful and 5 = not useful

6. Are the Problem Frame diagrams useful to trace requirements?

7. Are the AND/OR graphs of the KAOS method useful to trace requirements?

8. Please comment on the ease of writing a test script from the associated function specifications for:

(i) Problem Frame diagrams:

(ii) AND/OR graphs from the KAOS method:

9. Regarding dividing the problem into smaller parts, please comment on

(i) the Problem Frame partitions into simple problem frames:

(ii) the KAOS goal branching hierarchy:

Appendix I Recommendations and Guidelines

1	Focus only on the problem domain to elicit the requirements. Do not focus on the solution or start to build a solution until the problem is known. Recommendations are: • read available documentation on the problem domain; read any documents that the user has prepared describing their problem / requirement; observe the problem area and users; arrange meetings with all of the stakeholders. • questions to ask are: what; how; why; who; when. • the problem domain should be considered from an agent, a time and a case driven perspective so that all aspects are reviewed.
2	Create a high level model showing the boundary of the problem domain and all of its interfaces. Recommendations are a choice of: • problem diagram (more sophisticated notation than a context diagram so it can represent more information) • AND/OR graph showing the established goals. • context diagram

3	Decompose the problem into manageable parts and create sub-problem models. The decomposition can be hierarchical or made up of overlapping projections, the analysts' experience and preference might dictate this choice. The objective is that each part is not too complex to continue with. Recommendations are: • Projection into problem frames to be used when the sub-problems look to fit clearly into a basic problem frame: commanded behaviour; information display; required behaviour; simple workpieces; transformation. • AND/OR graph break down into goals to be used if the identified sub-problem does not fit clearly into a problem frame or the sub-problem is preferred to be defined in terms of system and private goals.
4	Use your financial environment's catalogued / classified past experience to see whether any requirement specifications can be reused.
5	Arrange meetings with all stakeholder groups to discuss the models and the break up into sub-problem areas. Use this opportunity to elicit information on any dependencies between the sub-problems.
6	If you are unfamiliar or do not have the required skill set required to analyse either specific functionality or technology then ensure that contact is made with a specialist or training is undertaken.

7	Gradually increase the level of detail until all low level detail is known. Recommendations are: • Problem Frames with full frame concern analysis. • Goal based sub-problems analysed according to the KAOS method. The possible ways of handling identified obstacles should also be provided Information on dependencies between sub-problems should be clearly highlighted. Regardless of the method chosen, each sub-problem should be reviewed for the following general considerations: • Overrun, the ability to respond to each external event • Initialisation, the initial state that the machine has to cope with • Reliability, the failure to fit a baseline requirement description • Identities, to ensure distinct identities can be identified where multiple instances occur and are not related to each other • Completeness, do the problem requirement, problem domain and machine description provide the full picture without ambiguities? • Obstacles that might prevent the sub-problem being solved • Constraints that need to be imposed on the sub-problem

8	Arrange meetings with all relevant stakeholder groups to discuss and agree the low level detail of the sub-problems. Take this opportunity to review whether any new regulations or functionality are imminent so that the adaptability of the requirements can be considered.
9	Create the final Requirements Specification document putting all individual sub-problems together so that a final solution can be created. All considerations from guideline 7 should be easy to see in the final specification document so that less is left to subjective interpretation. The final solution should now be considered so that all relevant information is provided in the specification. Where domains are in more than one sub-problem their separately defined characteristics should be brought together to ensure consistency. This will produce the first draft requirements specification and the document should now go through the usual review stages in line with other development lifecycle documentation.
10	Deposit your experience into your financial environment's catalogued and classified storage system for use in the future. A problem frames catalogue should be kept sectioned by problem frame. A goals register should be kept according to the classification system chosen. Possible choices are by target condition desired or by subject matter.