Chapter 2

2 Approaches to Knowledge Reuse

In Chapter One the ability to support users in a wide range of activities was seen as an important feature of KBS. This required knowledge to be reused and was called *activity-reuse*. In the previous chapter we also looked at the benefits of reuse in general and in this chapter we review current approaches to knowledge reuse which covers a number of types of reuse. These approaches not only differ in the solution they offer but also in the types of reuse they address. The main issues identified from the current knowledge reuse literature are given in Section 2.2. The focus on *activity-reuse* in this thesis is unusual, so a summary of what issues are considered relevant to this study is given in Section 2.3 at the end of this chapter.

2.1 Knowledge Reuse Research

Due to the many perceived benefits that can result from reuse of a resource there is currently much interest in the area of knowledge sharing and reuse. Weilinga et al (1993) consider reuse to be a common research goal of many current projects. Van de Velde (1993) terms ES with reusable components, second generation ES. Sharing and reuse was the theme at the 1995 Banff Conference on Knowledge Acquisition. Many believe that the building of large KBS will only be possible if efforts are combined (Neches et al 1991). Most of the reuse research discussed in this chapter is concerned with reuse of problem solving methods or ontologies. Section 2.1.2 introduces these research efforts. However, there is a body of research, known as the MYCIN experiments, which forms a notable exception and has been concerned with reusing knowledge for different purposes such as teaching or KA. This research is well known in the KBS community and has led the field in many regards including in the area of knowledge reuse.

2.1.1 The Mycin Experiments

The most well reported reuse of knowledge concerns the MYCIN family of ES (Buchanan and Shorltiffe 1984). MYCIN was originally designed to assist physicians in the diagnoisis and treatment of meningitus and bacterial infections. Its purpose was to provide a research vehicle to learn about and demonstrate the building of expert systems. The separation of the diagnostic process from the knowledge of the disease into EMYCIN (Empty MYCIN) was a demonstration of how problem solving knowledge could be reused with different domain knowledge. PUFF (Harmon and King 1985) is an ES which handles pulmonary disorders and is an example of a system using the EMYCIN shell that went into routine use. Numerous other adaptations of the original MYCIN system have been developed, these include:

- the reuse of MYCIN in NEOMYCIN to test that a knowledge representation (KR) could be used for explanation as well as problem solving.
- the abstraction of NEOMYCIN into HERACLES (Clancey and Bock 1985) for use as a heuristic classification shell. Clancey distinguishes HERACLES-DX as the first task-specific ES design (Clancey 1992, p.16).
- the development of TEIRESIAS (Davis 1977) to provide a debugging dialogue to assist the knowledge acquisition process for MYCIN. It did this by presenting a rule trace of its reasoning to the expert when a wrong conclusion was given. The expert could then determine if the rule was incorrect, the application of the rule was inappropriate or whether another rule could be specified to negate the incorrect conclusion. The addition of modelling tools to RDR to asisst with KA, described in Section 7.3.1.1 has a similar goal to that sought by TERESIAS. The RDR approach and TEIRESIAS both pass the responsibility for the system's learning onto the user, as opposed to many learning systems that seek to be self-adaptive. In the case of the RDR system described in this thesis, the emphasis on the role of the user in the learning process is intentional. In keeping with the system criteria discussed in sections 1.1.5 - 1.2.3 user control and system flexibility are seen as essential ingredients in gaining acceptance of computer systems.
- GUIDON-MANAGE is a tutoring program that uses subtasks to assist students to learn about diagnostic strategies using the knowledge in NEOMYCIN. In effect it acts as a simulation of using NEOMYCIN (Clancey, 1992). Clancey notes that when NEOMYCIN was reused in this way the subtask translations that were too abstract for consultation purposes became useful as hints for describing problem solving strategy.
- GUIDON-WATCH is another reworking of NEOMYCIN in a graphic form for instruction and debugging purposes. The system presents a causal model to the user and is a useful method of representation. This led Clancey to the concept of the situation specific model (SSM) which was also based on work in ABEL (Patil, Szolovits and Schwartz 1984). The SSM is the case model that results when a general model of problem solving is applied in a particular context. The SSM is manipulated by subtasks which are groups of metarules that perform part of a task. They can be considered as model construction operators (Clancey 1992), responsible for constructing nodes and links in the SSM. If they recur they should be abstracted for reuse.
- GUIDON-DEBUG was constructed to assist the detection of holes in the SSM.

The GUIDON systems mentioned above were developed as part of Clancey's PhD thesis to test the feasibility of the reuse of the MYCIN KB for tutorial purposes. Since then Clancey has had much to say on the topic of knowledge reuse. In his landmark paper on model construction operators (Clancey 1992) he claims "that describing knowledge acquisition tools in terms of model construction operators facilitates collecting and sharing knowledge bases and representation languages" (Clancey 1992, p.7).

From an analysis of numerous systems, including the MYCIN-related ones mentioned, Clancey lays to rest the fallacy that an ES consists of a knowledge base and inference engine alone. The third component is the inference procedure which provides the control or strategic knowledge. He says: "it is the ability to store facts about inference procedure constructs (subtasks, metarules, premise relations[7]) that enables it to be used for multiple purposes" (Clancey 1992, p.32). These procedures offer reusability of the abstract control knowledge. By employing different procedures the knowledge can be used in multiple ways:

> *Operating on a knowledge base in different ways (e.g. for explanation, tutoring, or compilation) requires different procedures with different relations, and hence a reclassification of expressions in the knowledge base" (Clancey 1992, p.10).*

By classifying process constructs into different groups the software can be reused by different interpreting procedures by offering different views of the model and reasoning processes. This involves establishing a new structure and new relations, which act as filters on what is included in the SSM. It is necessary to focus on the relationships between types of systems, process model structures, SSM structures and inference procedures.

It was also found in this thesis that abstraction, different views and understanding of the relationships between the primitive concepts in the KB (the rule premises and conclusions) were important to enable reuse. FCA is the means used in this thesis to automatically generate the relationships between concepts, to provide the structure of the knowledge and to provide a number of views of the knowledge. A major difference between the FCA work and Clancey's work is the absence in the former of metarules or procedural/control knowledge in supporting the different uses. The procedures described by Clancey in terms of subtasks and metarules are such things as Forward-Reason Generate-Question, Process-Finding Apply-Evidence-Rules, Explore-and-Refine, Process-Hypothesis. Instead of using these type of procedures to control manipulation of the domain knowledge the one inferencing mechanism is used on the low-level assertions and formal concept analysis is used to find the higher-level abstractions and relationships that exist.

---

[7] Subtasks are collections of metarules, a metarule is a rule that controls other rules and premise relations are the relationships between rule conditions.

Clancey sees other research into generalising as being relevant to reuse. Clancey equates Chandrasekaran's Generic Tasks to his definition of subtasks. What Clancey terms a complete inference procedure he equates to McDermott's role-limiting methods. Another point Clancey (1992) makes is that through the use of modelling tools and a system-as-a-model viewpoint (Newell 1982), an ES should be able to be reused for a range of different purposes, rather than needing to build different architectures for each problem situation. This thesis is interested in building a multipurpose system but, unlike most KBS approaches, the starting point is the development of an assertional KBS based on cases and observed expert behaviour rather than a terminological KBS which requires building high level models of the knowledge first. Before commenting further on the two contrasting approaches, let us look at the modelling perspective and generic approaches to reuse.

2.1.2 The Knowledge Level - Reuse of Problem Solving Methods and Ontologies

Like Clancey, many researchers stress the importance of the knowledge level (KL) perspective for sharing and reuse. The knowledge level was first introduced by Newell (1982), although, as Newell points out other research at that time implicitly acknowledged the KL viewpoint. An example is Marr (1982) who proposed a view using four levels that emphasized separation of implementation and computational decisions and analysis of expertise according to the tasks performed. Newell (1982) says knowledge has a competence-like structure and gives the following definition of knowledge as: “whatever can be ascribed to an agent, such that its behaviour can be computed according to the principle of rationality” (Newell 1982, p. 105). He argues that knowledge is described functionally not structurally. We need to know how knowledge works, why we need it and why it is effective to be able to use it. This is not an easy task. He points out, as AI researchers we are interested in generating knowledge intelligently. Simple procedures only generate uninteresting knowledge, a thought also expressed by Schank and Leake (1989), and will not be adequate for the second generation of ES. In reply to the debate over the best knowledge representation to use, he argues that logic is useful for modelling the knowledge level but it is not necessarily the choice for the symbolic level.

Prior to the introduction of the knowledge level, research had focused on the representation and manipulation of knowledge using symbols, also known as the physical symbols hypothesis (Newell and Simons 1976). The belief was that knowledge could be transferred or *mined* from an expert because both humans and machines required physical symbol systems to represent knowledge. Newells’ main idea is that in addition to the various layers that make up a computer system there is the knowledge level which sits above the symbolic top layer and has resulted in the abandonment of the knowledge-transfer attitude by most AI researchers. Van de Velde (1993) sees that separation of the knowledge (KL-Model) and the knowledge representation (symbolic level) is essential for reuse as it allows each aspect to be dealt with individually.

Van de Velde (1993) describes the following approaches which have been built on Newell's knowledge level model:

- Generic Task Framework (Chandrasekaran 1986)
- KADS and CommonKADS (Breuker 1994, Weilinga et al 1993)
- Role-Limiting Methods (McDermott 1988)
- Components of Expertise and the Componential Methodology (Steels 1993)
- Protege and Protege II (Puerta et al 1992)
- KIF and Ontolingua (Patil et al 1992)

Each one of these approaches are discussed below. Van de Velde describes these approaches as KL-models since they have more structure and are therefore less flexible than Newell's model (Van de Velde 1993). “A KL-model is a structure that is imposed on knowledge when it is being put to use in a class of problem situation” (Van De Velde 1993, p. 215).

While each of the approaches mentioned above are different, Van de Velde (1993, p.1218) considers three concepts to be generally included as part of a KL-model. These are the domain model, the task model and the problem solving method (PSM). Van de Velde argues that the task features and the domain model will affect the type of KL-model chosen and therefore it is important for more work to be done on how to select a KL-model. Selection of the right KL-model will affect the robustness of the system. The brittleness problem frequently referred to occurs when we build an ES to solve a particular set of cases and then try to solve an unseen case not previously covered by the knowledge base. To reduce some of the brittleness of ES, Van de Velde recommends matching the problem characteristics with the most suitable problem solving method.

Weilinga *et al* (1993) discuss the unification of the various knowledge modelling approaches. He particularly discusses CommonKADS, which combines the components of expertise and KADS approach. The suite of models in the CommonKADS methodology includes the expertise model, task model, communication, design, agent and organisation model. Weilinga sees human expertise as a separate model which may be useful input for the KBS expertise model but it is not the same, which matches Clancey’s (1993) criticism of descriptive models and is a view shared by Rademakers and VanWenkelhuysen (1993). In the same vein as Clancey’s argument for abstraction:

“The KL is the appropriate level for modelling the competence of KBS. It calls for the description of problem solving behaviour at a conceptual level that is independent from representation and implementation decisions” Weilinga 1993, p. 303.

Weilinga adopts the role limiting principle introduced by McDermott (1988). Role limiting refers to the way an agent imposes structure on the knowledge when faced with a particular task. Role limiting enables us to describe the ‘how’ of an agent in knowledge level terms. The role-limiting method requires the knowledge level to be broken into three categories - domain knowledge, task knowledge and inference knowledge. The latter may be equated to Van de Velde's problem solving method category. There is intra and inter-category structure. Domain knowledge includes the domain ontology and domain model. The domain model has its own schema and a more generic and abstract ontology which is different to the domain ontology. The model schema describes the structure of the entities in the domain model. Within these categories further generic types can be found. Problem solving knowledge is not a new category but a metalevel. It includes the PSM and strategic knowledge. Weilinga gives the following definition: a problem solving method “specifies a number of possible actions and some way of determining how these actions are to be ordered in time” (Weilinga et al 1993, p. 315). For a detailed description of each level see Weilinga et al (1993). The relevance of role-limiting for this discussion on reuse is that the idea of a task limiting and constraining the KA process means that the tasks handled by one role-limiting method will not be able to be reused by another role-limiting method so that reuse is only possible over the same task but not between tasks. As mentioned in the next chapter, recent RDR research questions whether it is possible to use the same problem solving method for different tasks such as classification and configuration and thus suggests that the role limiting method may indeed be unnecessarily limiting. Alternatively, it may be that classification is a specialisation of configuration and in this way they both share a similar role.

Weilinga et al (1993) distinguishes between what Newell described as the principle of differentiated rationality where we assume an agent will behave rationally to satisfy a goal and Van de Velde's two-step rationality where the knowledge configuration and application are also considered. Weilinga finds the latter to be a more useful description of how an agent will behave.

Weilinga et al (1993) describe a CommonKADS library of generic components which is being built to facilitate reuse. Generic components may be PSMs, instantiated PSMs, generic tasks, inference structures and generic domain models. Weilinga et al remark that some components may be too general (that is, the grain size is too coarse) and need adaptation. They are also developing a formal modelling language to assist reuse.

Other work building on the KADS structure uses the MoMo methodology (Hemmann 1993a and 1993b). Hemmann has a stronger commitment to reuse and sees that there are two separate types of knowledge that can be reused: domain knowledge and the problem solving knowledge. Unlike KADS, MoMo uses abstractions of PSMs, known as Frameworks of Problem Solving (FPS), which are said to be useful for knowledge acquisition, explanation generation, teaching, combining different implementations and the creation of repositories of different model types. He claims MoMo outperforms KADS in the area of explanation because of binding mechanisms which allow the user to be more involved in the inference process. Hemmann finds that KADS is "well suited to build reusable models of knowledge" (Hemmann 1992, p 1) due to the four layer framework and the widespread acceptance of KADS. While he argues that KADS builds models that are reusable, MoMo better supports sharing. MoMo is an operationalisation of the KADS concepts. In contrast to KADS, Hemmann (1992) has a greater emphasis on the separation of the domain layer from the rest, where a MoMo environment is made up of an interpretation model and a domain model. He equates the interpretation model to a problem solving method. Domain models can be used by different PSMs and a PSM could be used for different domain models. There is an emphasis on clear and clean links between the domain-specific and generic PSM knowledge. The joining of domain-specific and PSM knowledge is called a view (Hemmann 1993b).

Another approach is generic tasks (Chandrasekaran 1986). A generic task may be described in terms of the I/O description of the task, the method/strategy and knowledge. Generic task shells have been developed for classification (CSRL) and for design (DSPL). Generic tasks assist the process of knowledge acquisition by providing a vocabulary for seeking knowledge and guided organisation and use of the knowledge acquired. When domain knowledge is added the generic task shell works as a problem solver for the task in that domain.

Due to various problems such as chunkiness and inflexibility, Chandrasekaran has moved away from generic tasks to the task structure perspective. This structure is a tree of tasks, methods, operators, subtasks and inferences. Search-control knowledge and sub-task operators are kept separate to allow dynamic selection of operator sequence which offers wider applicability of the method to a range of problems. Methods are ways of organising knowledge to achieve a task. Methods are generic components that combine to form an organised collection of inferences for generic types of goals. Knowledge sharing continues to be a research concern for Chandrasekaran and Johnson (1993). They have come to the conclusion that knowledge can not be separated from its use and that different usages require different representations. This thesis questions this conclusion by demonstrating that knowledge can be used in different ways for different activities using the same knowledge representation. It is acknowledged, however, that the restructuring of the knowledge into the FCA subsumption hierarchy is another form of knowledge representation but since it is generated automatically we could view the multiple classification RDR and FCA representations as a general KR pair that together support a wide range of activities. The ability of the same RDR structure to support configuration and classification tasks also suggests, but is not substantiated in this thesis, that the same knowledge representation can be used to support multiple types of tasks.

In considering the use of generic tools, Van de Velde (1993) identifies the need to determine the type of problem to be solved. Breuker (1994) defines a problem as a discrepancy between the current and norm state (as opposed to the traditional view of a goal state). A solution is not a simple answer to the problem but a probably-abstract complex object comprised of a case model and an argument structure. The problem space can be described as <P,S,solution> where (P) is the set of problems, (S) is the set of solutions and (solution) is the set of relationships between P and S. Breuker points out that a problem is not the same as a task. In a task the problem solving methods are known and tasks can be repeated. Tasks are more structured, well-defined and general. Problems are deviations or complications. Tasks include three steps:- problem identification; problem definition; and task specification.

Breuker identifies eight problem types:- Modelling, Design, Planning, Assignment (scheduling and configuration), Prediction, Monitoring, Assessment and Diagnosis. These eight fall into two broad categories which he calls synthesis and analysis, more commonly known as construction and classification, respectively.

Breuker (1994) advocates using minimal solutions with generic conclusions. Rather than using a taxonomy to analyse tasks he prefers the idea of dependencies. In real life, task output (which is often a conclusion) from one problem is input to solve the next problem. For example a planning task may involve a chain of problems such as:

planning/assignment → prediction → monitor → diagnosis → assignment

Dependencies can be identified by distinguishing the problem from the task. Breuker suggests various common dependencies that are found. Breuker’s main argument is that within one application there is a suite of problem types and dependencies. He suggests that the set of problem types will be smaller and more similar than the set of tasks. He sees the development of a different problem solving method for each problem type as an ideal future goal.

Protege II (Puerta et al 1992) offers a similar approach by allowing knowledge engineers to select problem-solving methods from a library of domain-independent reusable resources. Protege-I was limited to a single method. The suite of tools assist in the definition of ontologies using MAITRE a frame-based editor for building MODEL structures, the building of KA interfaces using DASH and the development of run-time systems. The system uses the CLIPS production language and runs on the NeXT platform (Rothenfluh et al 1994). A domain ontology is used to map the library method to the domain. Ontologies and methods may be indexed and searched so that they can be edited and combined in numerous ways. Control knowledge is contained in the problem solving method and joined to the domain via the domain ontology that controls the reasoning process. This process of conceptualisation of the domain at a higher level and mapping to the problem solving method is one example of how the KL-model can be implemented.

Puerta et al (1992) describe Mecano, a component of PROTEGE-II, which reuses domain ontologies to build different knowledge acquisition interfaces. They argue that research has concentrated on the reuse of expert system problem solving methods and ignored the problem of having to build new interfaces for each application for knowledge acquisition. DASH uses a nodes-and-links graph depicting the dialog structure and the relationships defined in the domain ontology to infer the knowledge requirements, dialogue and presentation and layout. The benefit of this is that the relationships used in the KA tool is the same as those used in the problem solving method used for reasoning and results in reuse of the domain ontology.

The word ontology has arisen in a number of places and before going further it is worthwhile to define what is meant by an ontology as opposed to a knowledge base. The word ontology comes from the Greek words *ontos* and *logos* which mean, being and word, respectively, and therefore means that is it concerned with the study and description of being or the basic categories of existence (Sowa 1991). Although there are a number of variations, a common understanding of the word is a set of terms by which the domain or task can be described which includes definitions of the terms and the relationships between them. The definition of an ontology given by Gomez-Perez depended on whether the ontology was for reuse or sharing. If an ontology is designed for reuse it reduces the effort that would be required if it had to be built from scratch. If the purpose is sharing then an ontology acts as a common vocabulary and means of sharing one's viewpoint. Gomez-Perez (1994) offers the following differences and similarities between an ontology and a KB. The language used for an ontology needs to be "expressive, declarative, portable, domain-independent, semantically well-defined (and) machine-readable." (Gomez-Perez 1994, p.2). This language is not necessarily the target language of the application. The strategic and object knowledge should be separate in an ontology, although it is difficult to represent strategic knowledge declaratively. An ontology tends to be domain-independent and contains more abstract definitions than a KB. To enable reuse an ontology needs to be better structured and defined than a KB. "A KB is the knowledge module of a KBS. It contains abstract and specific knowledge of a particular subject in a machine-readable format" (Gomez-Perez 1994, p.2). Typically ontologies do not include reasoning methods although Guarino and Giaretta's (1995) definition includes predicates and functions that support operation on a logical language. Increasingly researchers such as van Heijst, Schreiber and Weilinga (1997) and Martinez-Bejar, Benjamins and Martin-Rubio (1997) use ontologies for the specification of knowledge-level conceptualisations. Thus the ontology provides a constrained structure on which the domain knowledge is based. Since ontologies are removed from the actual application in which they will be used it is very difficult to evaluate them. An exception is the work described near the end of this section (Ikeda et al 1996) where the conceptual model can be executed at the symbol level which allows ontological commitment to be validated. This question of evaluation is discussed later under section 2.1.3.1 which covers the problems with generic solutions.

While there is little consensus on a single definition of an ontology the motivation for the development of an ontology and the type of ontology provided by the FCA line diagrams in this thesis is in keeping with the general "aim of ontology research [which] is to explicitly represent the meaning of concepts and the relation among them" and the notion that an "ontology, in general, is an agreement between users and systems" (Ikeda et al 1996).

Returning to a review of KL-model approaches, Steels (1993) advocates breaking the KB into components, however his criteria for grouping is not knowledge content but tasks, domain models and methods. This he calls the *componential framework*. The task view looks at what needs to be achieved, the model views looks at what type of models are being built and utilised. The method view considers how to use the knowledge to accomplish the task. A brief description of models and methods and how they relate to tasks are given below. Refer to (Steels 1993, pp.276-281) for more detail.

A model that is fixed is known as a domain model. A dynamic model is a case. A domain model is used to solve a specific case model. He refers to ontological models as special cases of domain models that "constrain the vocabularies that can be used in other models" (Steels 1993, p.276). It is due to this inflexibility that Schmalhofer, Aitken and Bourne (1994) criticise the ontological approach to reuse and their solution is offered in section 2.1.3. A model is related to a task either as a target model, which is affected by a task, or as a source model which is consulted by a task. Model dependency diagrams are used to show inputs, outputs and interfaces. He breaks tasks into domain acquisition, application, solution and decomposition tasks.

A method is an algorithm that organises a series of activities. Task decomposition methods break tasks into subtasks and solution methods define how to perform the subtask. The decomposition grainsize will affect the size of the reusable chunks. These methods can be further subdivided into acquisition, inference and presentation methods. All of these are of interest to reuse. The acquisition and presentation methods affect the way the user is able to interact with the system. Inference methods include learning methods and problem solving methods. Problem solving methods are concerned with case models and the question of 'what'. Knowledge is more concerned with 'why' (Clancey 1992, p.24). Learning methods enable new domain models to be inferred.

Steels comments that it would be impossible to provide a workbench which includes all of the methods because of the volume of methods. Even if possible, selection of the appropriate method would be untenable. He envisages a workbench with a range of methods that suit a particular application domain.

Steels(1993) describes COMMET as such a workbench which has been created to allow experimentation with reusability. The application is built at three levels: the knowledge level, execution and code level. The application is first designed to produce a knowledge level model in terms of tasks, models and methods. Next the system is built at the execution (the implementation of the knowledge level on computer), followed by the code level (the actual files). The complete application with the three levels is called a project and can be used to acquire knowledge. To specifically assist the developer with reuse of components there is an application kit manager which can query and guide the developer to the appropriate chunk. This chunk can then be adapted and fused at either the knowledge or execution level. COMMET has provided more structure for the design of a KBS and the acquisition of knowledge. Its ability to facilitate reuse of KBS components is of interest to this study but not directly related since the focus of COMMET is not on *activity-reuse*.

There is another area of research which combines the ontological and problem solving method approaches by developing an environment for building ontologies of problem solving methods or otherwise known as task ontologies (Mizoguchi, Vanwenkelhuysen and Ikeda 1995, Ikeda et al 1996). The implemented system is known as the Conceptual LEvel Programming Environment (CLEPE). Ikeda et al (1996) acknowledge that mainstream ontology research focuses on general ontologies and their reuse on the assumption that the human does most of the work involved in the task. They take a different view based on the premise that "where a machine plays a central role in a task .. we may say that task ontology should be investigated first". They elaborate that the task ontology should be a first principle of knowledge engineering where the "machine is intelligent enough to cooperatively work with a human and computationally strong enough to play a central role in problem solving" (Ikeda et al 1996, p. 211). CLEPE provides a Task Ontology representation Language (TOL) together with a tool for editing and browsing the ontology. Generic Process Networks (GPN) are used to offer a lexical level ontology which represents the user's problem solving process. GPN are made up of a natural language component and a task ontology translation component of the process. Using GPN users are able to use natural language to describe the task which is then translated by CLEPE. The conceptual model is executed and shown to the user where differences between what was intended by the user and what has been developed by the system can be reconciled. The purpose of this step is to ensure ontological commitment. This research differs from other research mentioned in that it is a hybrid ontological and PSM approach which focuses on the development of task rather than general ontologies. The role of the user in the specification of these models and the ability to execute the model for validation by the user are further features which distinguish this work from the mainstream. While the mapping between the conceptual (knowledge) and the symbolic level is well defined and avoids one of Schmalholfer, Aitken and Bourne (1994) criticisms of the Knowledge Level mentioned below, this work still fits with mainstream KA research in so far as modelling is seen as a prerequisite to KA.

This section has described the mainstream approaches to KBS development and reuse. Despite the widespread acceptance of the knowledge level as the basis for KBS development and the trend towards generic solutions there are a number of limitations associated with these approaches. The next section includes an analysis of the why the knowledge level has such limitations and how the work in this thesis is aimed at avoiding such shortcomings.

2.1.3 Limitations of Knowledge Level Approaches and Generic Solutions

The approaches described in the previous section have been based on modelling at the knowledge level. In keeping with the arguments offered in Chapter One of this thesis, the ability to reuse knowledge has been seen by many as essential to support the development of large KBS and many research efforts have developed generic approaches to support reuse. Unfortunately, there also appear to be a number of shortcomings of the generic and knowledge level approaches that various researchers have identified. In this section we look at these limitations, including a subsection which considers how the knowledge-level approaches are being evaluated and validated. In the final subsection, we consider how the approach used in this thesis may overcome some of these weaknesses.

The knowledge level is based on the assumption that certain actions will be performed by rational agents according to the given knowledge and goals. This is not necessarily so and does not differentiate adequately between different possible situations and when a PSM should be applied. Breuker (1994) pointed out that most problems will require a suite of problem solving methods. Selection and combination of this suite is difficult (Zdrahal and Motta 1995) and made more difficult by the situated nature of problem solving. What one person perceives as the solution to a problem will differ in each situation with what is perceived by another person. Even what constitutes a problem in the first place will vary (Lave 1988, Schon 1979).

Schmalholfer, Aitken and Bourne (1994) describe four misconceptions upon which the Knowledge level is based:

"*1. Knowledge and goals are in themselves inadequate to fully characterise intelligent systems.*
2. *Knowledge level descriptions are developed as if intelligent systems were causal systems.*
3. *For this level of abstract description, the distinction between an agent and its environment is artificial.*
4. *The knowledge level does not lie directly above the symbol level and there is no tight connection between them. Therefore knowledge level descriptions can not be reduced to the symbol level (Clancey 1991)* " (Schmalhofer, Aitken and Bourne 1994, p.87).

They further cite Clancey's criticisms concerning the frame of reference problem (Clancey 1991a) which basically argues that a description of a model is not the same as the model and that definition and interpretation of that model will require entering inside the agent so that the situation is framed in the same way. Another problem is that as systems are embedded in larger systems their appropriateness changes. The knowledge level does not accommodate change and results in knowledge level ontologies that are static and difficult to reuse (Schmalholfer, Aitken and Bourne 1994). To address the lack of flexibility of many generic approaches, some research groups are looking at smaller-grained reusable components (Tu et al 1994). Schmalholfer, Aitken and Bourne (1994) propose an extension to the Knowledge Level model using behaviour level descriptions that take into account goals, knowledge, skills and performance and the relationships between them. Context is important in allowing sharing and reuse of behaviour descriptions as they can be adjusted to support new uses.

As mentioned in Chapter One, the focus on the knowledge level has resulted in a focus on knowledge elicitation rather than user requirements elicitation. Salle and Hunter (1990) point out that the two differ and ideas and opinions form part of user requirements. So while they acknowledge the usefulness of the knowledge level, Salle and Hunter recognise that its usefulness is limited by how good a model can be found. A major problem with specifying user requirements is that users often do not know what they want or need. So the elicitation of knowledge and user requirements both suffer from the situated and uncertain nature of human knowledge and action.

Gennari et al (1994) argue that the original goal of reuse to reduce the KA bottleneck may introduce new bottlenecks if the domain and methods cannot be easily and appropriately connected. There are two different entities that may be reused: a domain-independent problem-solving method or a domain description that can use multiple methods. If the aim is to reuse problem-solving methods and ontologies then there needs to be a mechanism to join the two. Gennari et al (1994) have explored the use of bridging elements called "mapping relations", which become the third entity in the reuse kit. As they point out, the development of reusable components is a challenge since methods often become intertwined with the domain.

While there is widespread support for the development and use of generic KBS solutions. Rademakers and Vanwelkenhuysen (1993) describe some of the problems that can occur with the use of generic models. They define generic models as "predefined partial or abstract models about tasks, problem solving methods or domain ontologies" (Rademakers and Vanwelkenhuysen 1993, p. 354). As they point out generic models can assist as templates for top-down knowledge acquisition and as skeletons for abstracting raw data. The problems occur because the knowledge level is a model and is therefore subjective and imprecise. In fact we are dealing with two models that need to be kept separate, the expert's problem solving behaviour and the intended system behaviour.

Some of the problems they identified are:

1. Multiple generic models apply, but with different degrees of appropriateness.
2. Generic models are often too general to give useful support and guidance.
3. Generic models can seldom be used 'as is' and may need to be adapted or combined.
4. The terminology of generic models may be hard to tailor to the audience to which it is communicated. Matching between audiences and model terminology will be necessary.
5. The consequences of selecting a wrong model can be very costly if it is found that it does not support the needed functionality and a change to another model is required.
6. The intended goal will affect the appropriateness of the model.

7. Viewing knowledge acquisition as a knowledge transfer process misleads the knowledge engineer into focusing solely on the human expert. Context, expectations, KE experience and an experts tendency to justify rather than explain need to be considered.
8. Important details may be overlooked when trying to fit the generic model.

Despite these shortcomings, Rademakers and Vanwelkenhuysen describe some types of generic models and when they may apply. The main distinctions they make between approaches are based on the kind of generic model and the model's granularity. They suggest the following guidelines to reduce the problems identified above:

1. provide support for mixed modelling,
2. distinguish between observable expert behaviour versus desired system behaviour (current generic models concentrate on the system),
3. modeling expert problem solving behaviour alone is insufficient, it is necessary to consider communication and cooperation between actors,
4. a model of expert behaviour positions or orients the consultant to understand the behaviour - the environment should be accounted for and biases of experts avoided,
5. a model of expert behaviour is a source of information for system specification and design,
6. domain characterisation should precede the model construction phase,
7. models should be simple and transparent to its users,
8. modelling vocabularies should be extended to cover cooperation, negotiation and communication,
9. modelling and validation should be tightly coupled.

The last point is taken up in the next subsection.

2.1.3.1 Validation and Evaluation Concerns

The approach to verification and validation (V&V) of Knowledge Sharing Technology (KST) is not clear and there is not a great deal of energy being expended in this area. Gomez-Perez (1994) finds this situation has occurred due to a vicious circle of no standards so no evaluation, no evaluation so no need for standards. Hemmann (1992) argues widely accepted standards are necessary if reusability is to be possible. Gomez-Perez states that if KST is to become useful it is necessary to have some guidelines of when to use them, how to validate and verify them, where and how to find resources and how to use them. Gomez-Perez suggests the use of competency questions or counting the number of applications that successfully reuse the ontology. He provides an extensive table of approaches to evaluation by other researchers. Gomez-Perez offers detailed definitions of V & V specifically related to ontologies and adds the category of assessment which is how useful and usable a system is. Verification is checking that the system is behaving in the way it was designed. Validation is checking that the system is appropriate for the problem.

Rothenfluh et al (1994) see the use of projects such as SISYPHUS (Shadbolt 1996) as a way of evaluating general architectures. They argue that any architecture must be tested with real-world tasks. This thesis has taken advantage of the material supplied as part of the SISYPHUS III experiments in the geology domain and the usage and results are described in Sections 5.3.3, 6.4 and 8.2.2.3. Such collaboration with other research groups tests the robustness of approaches and allow comparison. A project specifically aimed at KST could uncover and address some of the issues in V&V of such technology. The commonsense KB developed by DARPA is being tested in this way.

Biggerstaff's "Rules of Three" (Tracz 1988 in Hemmann 1992) may be a useful guide when deciding when reuse of a component is useful:

1. *Before one can develop reusable software one needs to have used it three times.*
2. *Before one can reap the benefits of reuse, one needs to reuse it at least three times.*

In an evaluation study of PSM and their usefulness for reuse, Menzies (1997b) has concluded from the PSM reuse literature that PSM change is more frequent than PSM reuse. He also argues that the general lack of consensus between PSM research groups on the number and nature of PSMs and a set of PSM primitives, such as select or classify, and even on the definition of how to perform the same task, such as diagnosis, are all indicators that the use of PSMs has not stabilised and therefore the reuse of them is unlikely.

Menzies (1997b) cites two studies that have provided some statistics on PSM-based reuse: SPARK/BURN/FIREFIGHTER (SBF) (Marques et al 1992) and Mechanisms for KA (MeKA) (Runkel 1995). In SBF, the user supplies business information which is used by SPARK to create a domain-specific KA tool. A structured interview is conducted by BURN which maps the business information into a library of sub-routines (known as mechanisms). Meta-knowledge in the PSM is used to guide the mapping process and to determine what questions should be asked from the user. When the process is complete a rule-base is generated which is used by FIREFIGHTER to assist the user to execute and debug the performance system. In MeKA, each mechanism corresponded to a PSM. The PSM was broken into data structure and control knowledge and contained four modules to support acquisition, verification, generalisation and a dialogue module. The purpose of the study by Runkel was to see how frequently each

PSM developed using MeKA was reused in the development of subsequent systems. Table 2.1 shows the results of eight applications that were developed. The first application in Table 2.1 shows no reuse since all MeKAs were new but the other seven applications range from 50 to 88 percent reuse.

App. No.	Application Name	MeKAs used/total MeKAs
1	Room assignment	0/7 = 0 percent
2	Elevator configuration	5/8 = 63 percent
3	Elevator design validation	7/8 = 88 percent
4	Configuration validation	6/7 = 86 percent
5	Truck design 1	5/9 = 56 percent
6	Truck pricing	10/12 = 83 percent
7	Truck design 2	8/16 = 50 percent
8	Truck manufacturing	15/17 = 88 percent

Table 2.1 An evaluation of Mechanisms for KA Reuse
(In Menzies 1997b from Runkel 1995)

The 13 mechanisms included in SBF have been used to build nine applications that showed an decrease in development times from 1 to 17 days (using SBF) to 63 to 250 days (without SBF). Menzies (1997b) sees this result as significant but points out that from the published reports it was unclear whether the applications built with and without SBF were of equal quality and whether the personnel using SBF were the SBF developers or others biased to SBF. Another limitation is that experiments on such a scale are difficult if not impossible to repeat because they require a substantial commitment of resources which result in throw-away systems.

The MeKA study has a greater likelihood of being repeatable but once again it is unclear whether the favourable results are due to the skill of the developer or the usefulness of the tool. It is difficult to compare the two studies because the SBF study has used a reduction in development time as its benchmark and MeKA has used the amount that mechanisms were reused as the evaluation criteria. A higher percentage of reuse may be an indicator of an increase in productivity and reduction in development time but this is not made clear.

What can be argued from these studies, and is in fact what Menzies (1997a) argues, is that support for the reuse of PSMs is not clear and the benefits described in these studies can be seen as support for standard reuse of code and data structures that we see in software engineering. Menzies cites Stark (1993) who reports 70-80 percent code reuse using FORTRAN and object-oriented (O-O) design principles. Frakes and Fox (1995) consider more than code and found maximum median values for reuse in requirements, design, and code reuse at 15, 70, and 40 percent, respectively (Frakes and Fox 1995). It is interesting that:

> *"Frakes and Fox found no significant correlation between reuse and technology options such as the use of CASE tools; the presence of code repositories; or language level (assembler has a lower language level than O-O languages such as Smalltalk). Instead, the factors that were positively correlated to reuse were all organisational factors such as reuse training, unified software process; or industry type (for example reuse was high in telecommunications industries, which was possibly facilitated by the use of standard hardware configurations in that field).Menzies 1998, EVAL-5 p. 13.*

Menzies makes the point that the reports of Stark (1993) and Frakes and Fox (1995) suffer from inexact measures of reuse and that PSM researchers should be seeking to test whether the use of PSM results in benefits over standard reuse of software. We now return to an overall analysis of knowledge level approaches.

2.1.3.2 An Analysis of Knowledge Level Limitations

It is not the intention of the author to imply that there is no merit in general problem solving and ontological approaches to reuse. The point being made is that they have sufficient limitations to warrant consideration of alternative approaches. This thesis argues, and demonstrates, that complex descriptive models are not necessary prerequisites for building KBS.

Many of the problems identified with knowledge-level PSMs, ontologies and generic models are related to the inherent limits of descriptive models and the effort needed to interpret those models when they are applied. This situation is summed up by Chapman when he states:

> *"Representation is generally thought to make problems easy; if a problem seems hard, you probably need more representation. In concrete activity, however, representation mostly just gets in the way" (Chapman 1989, p. 48).*

A useful description of the types of models that are involved in capturing conceptual models is provided by Shaw and Woodward (1989). It is possible that experts have their own internalised model known as the mental model. When the expert articulates that model it becomes the conceptual model and the model that is developed from that is a model of the conceptual model, not the internalised model. To add to the difficulty in achieving a model that is truly representative of the expertise, the mental model may not correspond to how the problem is really being solved, the expert may be unable to articulate their mental model into a conceptual one and the knowledge engineer may not interpret or represent accurately the conceptual model into the final developed model.

Norman (1986) explains that when the developers conceptual model of the users mental models do not match, the design model will conflict with how the user expects to use the system. The key is to make the mental model explicit so that it is clear what the design model should be. This is an argument for building KBS using the RDR approach, that is capture knowledge in the way the expert exercises their knowledge and then look at the models that underlie the knowledge. This is another way in which knowledge in an RDR KBS becomes a mediating representation taking the user from what they do to how they think.

Menzies summarises what he believes to be the major potential and actual benefits of reuse in a very interesting paper which makes comparison between the lessons learnt from ES software development and the use (and reuse) of patterns[8] in the object-oriented community. Menzies (1997a) describes the three potential benefits of reuse of patterns: the reuse benefit which lets a designer build systems more efficiently using proven old systems; the guidance benefit that provides insights discovered in other systems and the communication benefit where patterns are useful in explaining existing systems. Menzies (1997a) sees that adoption of the knowledge level has brought with it the benefits of a more structured approach that has resulted in better organised projects that have seen some industrial success. These are guidance and communication benefits. Menzies (1997a) doubts that the reuse benefit of efficiency has yet been achieved. The KL approach approach is superior to the transfer-of-knowledge approach and acknowledges that knowledge is a model and not an artifact. However, the amount of effort being put into getting the model *right*, in such forms as PSMs or ontologies, appears to treat the model as an artifact and does not adequately acknowledge the deficiencies of models. A major concern of this thesis is that the preoccupation with complex models as a prerequisite to KA produces a bottleneck of its own with KA itself becoming a side-issue or afterthought. A situated view of knowledge does not support such an emphasis on developing good models but demands that systems are grounded in the real-world and support incremental change.

It may be that the focus on the development of complex models and analysis has occurred because much research has been based on laboratory experiments which have shown the regularities between expert behaviour and the use of reflection and abstraction. However, such settings are missing the broader context and many of the features and constraints that typically affect expert behaviour in natural settings (Lave 1988). Laboratory settings usually involve an individual in a predefined task that has

---

[8] "A fragment of a high-level conceptual model which may be useful in many applications (Menzies 1997j) oo-patterns

already been analysed and the problem-space has already been structured. The results from these studies characterise expert behaviour as reflective making use of “abstract, accepted methods that constrain the activities of practitioners and render the activities of practitioners predictable” (Shalin et al 1997, p. 195).

Given the limitations of laboratory experiments and the situated nature of knowledge, we may question the feasibility of building KBS to be used in the real-world. However, Shalin et al (1997) go on to report studies of behaviour of real-world tasks which show use of accepted methods but the decision of when to employ which method is based on “perceptual experience and temporal awareness obtained through engagement with the physical world” (Shalin et al 1997, p.196). Shalin et al (1997) have studied some contextually rich environments that cover what Woods (1988) calls “complex, dynamic and physical domains”. Such domains often involve large systems, multiple experts, quick turn-around from problem to solution and critical decision making where a wrong decision may have direct or indirect catastrophic results. The good news is that although at the low level the domains may be “complex, ill-structured and unpredictable” at some abstract level human behaviour can be seen as predictable. These predictable actions can be seen as stable behaviours (Clancey 1991b). Shalin et al (1997) state: “because accepted methods with well-defined conditions of applicability appear to dominate performance in these domains, we are optimistic about the viability of artificial aids for certain functions ordinarily thought to be intelligent” (p.213). Collins (1987) also comes to the conclusion that it is viable to use symbolic systems but we must acknowledge the role of the human in filtering the input and output accord to their social context. Similarly, while Suchman (1987) showed that plans are adapted to fit the situation and the adaptations are executed reflexively, she does not argue “don’t make any plans”. Plans are useful but we must be able to change them.

The RDR paradigm has been used in domains that qualify as “complex, dynamic, physical” and have been built not only to support but to encourage continual adaptation or refinement. Instead of the complex analysis route, RDR captures a simple model focused on the observed stable behaviours of the expert. The role of the user is paramount in building, maintaining and using the system and the expert acts as the filter that controls the inputs and outputs supporting customisation of the knowledge to suit the particular environment (Edwards 1996).

This section has described a number of knowledge level approaches and generic solutions to knowledge reuse. These approaches have been concerned more with the development of general reusable problem solving methods and ontologies than with reuse of domain knowledge which is viewed as more specific to an application. There is an implication that once the model is right the domain knowledge can be acquired fairly straightforwardly. This, however, is often not the case. In the next section we look at an alternate group of researchers that are focused on the capture of commonsense knowledge.

2.1.4 Reuse of Commonsense Knowledge

Many of the research efforts described above have been primarily concerned with the sharing of knowledge base components, problem solving methods and tools rather than sharing of the domain and lower level knowledge. Two notable large-scale exceptions are the DARPA Knowledge Sharing Effort (KSE) (Patil et al 1992) and MCC Corporation's CYC Project (Guha and Lenat 1990).

The KSE has a vision of providing a library of KB resources, which will reduce effort, cost and time and allow the building of large KB systems. The KSE is looking at four main areas:

- The Interlingua Group are creating a language (Knowledge Interchange Format KIF) that allows heterogenous knowledge representation languages to interchange knowledge. KIF enables systems using different KRs to communicate. To do this KIF accepts the sending KR format, converts it into KIF and then translates from KIF into the receiving KR. They see this solution as a better alternative to writing translation programs for every possible combination of KRs.
- The Knowledge Representation System Specification (KRSS) Working Group is concerned with variations or dialects within knowledge representation families. They aim to specify a common version of each main KR family providing a practical and comprehensive version which includes the best features.
- The External Interfaces Working Group are looking at establishing conventions in the area of communication between KBs and other computer technologies such as database, hypertext, accounting packages, etc. They have developed a Knowledge Query and Manipulation Language (KQML) which includes a protocol and common high-level language for information and knowledge exchange.
- The Shared Reusable Knowledge Bases (SRKB) Working Group is concerned with creating libraries of reusable and shareable knowledge modules. Ontolingua has been developed to provide a portable representational vocabulary which allows sharing and reuse of ontologies (a domain of discourse).

The last group is of greatest interest to this study. The focus of the SRKB is on the content of the knowledge captured rather than the aspects of how to represent the knowledge or share the knowledge, which are the concerns of Interlingua and the KRSS groups, respectively. As noted in section 2.1.2 there are a number of definitions of the word ontology used within the AI community and the SRKB have their own. The SRKB see a common ontology as:

> *"a set of definitions of representational terms used to construct expressions in a knowledge base, such as classes, relations, slots and object constants" (Patil et al 1992).*

This becomes a shared vocabulary with shared definitions. The research has been broken into the following three models. The library model is concerned with making a range of knowledge bases available and reusable. The software engineering model includes the sharing and reuse of software components. The reference model involves providing ontologies for describing domains or problem areas.

The other significant effort is MCC Corporation's CYC Project (Guha and Lenat 1990). The approach of this group is to build a large commonsense KB that can be used for building various applications. This avoids the problems of heterogeneous KR systems, family dialects, and different communication protocols. Over one million assertions have been established through various collaborations with other researchers (Soloway, Bachant, and Jenson 1987, Cohen and Loiselle 1988 and others). The KSE group see the CYC project as complementary and are providing compatibility with the CYC-L KR language through KIF. In this way the CYC outputs become components of the KSE library of resources. Over time, as sharing becomes widespread the CYC and KSE visions may lead to the same result.

Research on a smaller scale is being conducted by Pirlein and Studer (1994) who have developed KARO (Knowledge Acquisition environment with Reusable Ontologies). Their definition of an ontology is broader than the one presented by the KSE definition. They refer to that definition as a representation ontology, one that provides structure but no guidance. They are more concerned with what they term commonsense ontologies where an ontology is "the topmost level of a large hierarchy of commonsense knowledge" (Brachman 1990, p.1089). They have built a commonsense KB using the LILOG ontology with 700 concept definitions clustered into groups covering objects, time and events, qualities, quantities, measuring scales, energy and motion and assemblies and matter. LILOG is a colloboration between four German universities and IBM Germany. They use a formalised engineering framework which supports the KADS domain, task and inference layers called Model-based and Incremental Knowledge Engineering (MIKE).

Pirlein and Studer claim superiority to the KSE Ontolingua approach because KARO assists in the creation of new conceptual models of a domain by providing tools and methods rather than simply providing a communication medium between domains.

These projects, particularly the KSE and CYC, are more concerned with the concept of sharing between groups and across applications than reuse. In contrast, the focus of this study is on reuse rather than sharing. The distinction may seem spurious because the same benefits of reuse are being gained. In either case knowledge only needs to be captured once and can then be made available to a wide audience. This reduces the amount of effort required and also brings the added benefit of consistency. However reuse is more concerned with the aspect of different uses rather than the same uses in different applications. Commonsense knowledge can be seen as very low level knowledge not specific to a particular domain or application and involved sharing of the concepts across applications. The knowledge needed to support *activity-reuse* is more concerned with reuse of the actual domain knowledge within the same application possibly by the same people to support different decision situations and styles. Work by Puerta et al (1992b) does look at offering multiple interaction styles using Mecano which is a component of PROTEGE-II. PROTÉGÉ-II supports reuse of ontologies and problem solving methods which can be combined and presented to the user in different ways. However, the focus of Puerta et al (1992) is on developing knowledge acquisition tools from domain ontologies to suit a range of needs. The focus of this thesis is broader than simply knowledge acquisition.

The research by Steels (1993) and other work by Marques et al (1991) is more concerned with reuse of knowledge base components rather that the knowledge itself. If we think in terms of software engineering, much software reuse research is interested in the reuse of programs, procedures and data definitions (which can be equated to the PSM and ontologies) rather than the data (the domain knowledge). The sharing of knowledge and the reuse of components are concerned with reducing the effort involved in building KB and thereby enabling the development of large KBS. The focus of this study is not on reducing the effort in building a KB, but on how to build a knowledge resource that will encourage the use of the resource and allow the gain to be maximised. There are two reasons for this emphasis on reuse.

Firstly, it is debatable how willing organisations and individuals are to share their knowledge with others. The advent of the web has not significantly changed this and industrial use of the web tends to be for marketing purposes with possible free samples to download as an enticement to purchase a full system. With the exception of companies whose business it is to offer public databases, we generally do not see organisations sharing their databases with other organisations. On the contrary, corporate databases are strictly guarded from unauthorised and external access. How much more important is a company's knowledge resource, which contains not only data but valuable expertise as well ?

This is not to say that the KSE and CYC efforts are a waste of time. The development of a commonsense KB that acts as the foundation for building other domain specific KBS is highly desirable. Without such an effort no individual organisation would be able to build and maintain such a resource. The standards set by the KSE will be of benefit to the AI community. However, as with all standards, particularly true of the diverse computer industry, the difficulty, is in having them adopted. Typical of this problem, KSE and CYC have developed their own ontologies. Chandrasekaran and Johnson (1993) ask:

> *"is there any sense in which we can determine and agree on a set of terms that are not idiosyncratic to each research paradigm, but in fact represent some universally sharable set of analytic primitives ?..... We need a principled theory of inference operators and how they connect to world models that we construct and the tasks that we perform..... theories of generic tasks and methods will eventually have to be grounded in such a categorical content theory of our thought" (pp:267-268)*

On the other hand the 'holy grail' sought by many of a unified approach may not be terribly practical. As a study by Gambetta (1995) points out, experts prefer to use their own terminology. In the area of knowledge representation the KSE has taken the approach of offering compatibility to a range of KRS in contrast to the CYC effort that have committed to one representation.

Having looked at a number of approaches to knowledge reuse we consider what issues appeared to be the most relevant to the reuse of knowledge in general. Following this we look at the issues specific to *activity-reuse*.

2.2 The Issues Relevant to Reuse

Some recurring themes are apparent from the discussion of the reuse literature above. These themes include the importance of contextual, *deeper* and more knowledge and the use of levels of abstraction. These issues are discussed further in sections 2.2.1 to 2.2.4 which follow. Section 2.3 provides a discussion of the issues that appeared most relevant to the work in this thesis.

2.2.1 The Importance of Context

The issue of context is becoming an important factor in the reuse and sharing community (Chandrasekaran and Johnson 1993 and McCarthy 1991). Research is being done by the knowledge sharing effort (Patil et al 1992) into adding contexts into Knowledge Interchange Format (KIF) to facilitate the translation of facts from one context to another. In the area of natural language, context plays a major role as the context of a word will often determine its meaning. In answer to the growing problem of generality in AI, Guha under the supervision of McCarthy began looking at the need to provide context. Large KBS such as CYC require context to be captured, as well as knowledge, so that the knowledge could be applied to the different domains covered by the system. If knowledge is to be shared the context in which it was developed needs to be understood and described, an argument made earlier by Clancey (1992), so that the knowledge can be adapted to fit the new situation. Without consideration of context, attempts to collect commonsense knowledge "like so many butterflies" would be advocating a return to the 1960s where the AI community viewed "memory as storage" and believed "knowledge is power" (Clancey 1991b, p.245).

McCarthy (1991) and Buvac and Mason (1993) have developed a formalisation of the context concept. They argue that context allows a specific system to be developed and later generalised to fit the new problem since the context acts as a lifting axiom which transcends the original context. To support this a new modality has been introduced:

ist(c,p) where ist = is true , c = context and p = proposition.
Meaning that the proposition p is true in the context c.

The situated view places even greater emphasis on the role of context. If we believe Rosenfield who states that categories are not "stored things" but that "categorisation occurs at runtime" (Clancey 1991b, p.243) it is hard to understand how ES have found any success, particularly in the area of classification. The ability of experts to state some rules that can accurately predict to which class an object belongs is because classification can be considered a stable behaviour (Clancey 1991b, p. 249). The issue of context is highly relevant to stable behaviours. If we consider the phenomena of only being able to remember a person's phone number whilst in the process of dialing that number (Norman 1988), stable behaviours may be context dependent since they tend to become subconscious or reflexive actions.

When considering context, we must consider the environment, not just the general model and SSM (Clancey 1992). When knowledge is to be reused in different environments it is necessary to consider what is general to all situations. Consideration must be given to what is the case population, what are the world data assumptions (biases), model usage, interactional data and communication constraints. The assumptions about populations, model content and environmental boundary conditions need to be captured in the knowledge base so that they can be changed or tested against when a knowledge is reused for different purposes/situations. Clancey (1992) suggests the use of a matrix of systems, modelling methods and communication environments for boundary testing.

Contextual knowledge extends beyond environmental factors and also includes emotional, social and historical factors. Environmental and historical factors seem to be the aspects of context that most reuse approaches are able to support. Social factors have a high impact on the relevancy of knowledge, what can be considered expertise and affects not only groupware but also single user systems. The ability to customise medical systems to individual environments and cultures is a feature that Edwards (1996) comments is essential for actual systems and is offered by RDR since system development and maintenance is easy and performed by the user/s. This allows tailoring of the knowledge to the social and environmental context.

Emotions appear to be the most difficult aspect of context. On the one hand, emotions colour our perception and affect our recollections. On the other hand, the systems we build should give consistent recommendations. This does not mean that we should ignore emotions but that emotions pose too much of a problem and the alternatives are not clear. The typical approach to emotions is to treat them as a variable that affects the other inputs or outputs depending on the value assigned such as happy, sad, bored, etc. However, this type of approach does not capture the unexpected aspect of emotions which could make the same person act one way one day and a completely different way another day in what appears to be the same circumstances. While Karunananda (1993) still essentially treats emotions as an attribute, Karunananda has taken into account the cumulative effect of emotions which is often the cause for different responses to similar stimuli. Her approach is based on Buddhist philosophy of the human mind.

Context is an important aspect of RDR which is captured in the cornerstone cases and the exception structure. In section 3.2.2 context is discussed in some detail together with a description of how it is handled by RDR. We move on to the next issue which is the abstraction of knowledge.

2.2.2 The Abstraction of Knowledge

Many have advanced the view that reuse is facilitated by structuring the knowledge into different levels of abstraction (Chandrasekaran and Johnson 1993, Clancey 1992 and Swartout and Moore 1993). Hemmann (1992) points out that this is a lesson that should have been learnt from the experience of reusing software.

Chandrasekaran and Johnson (1993) offer an interesting notion of what abstraction is all about. They argue that knowledge needs to be normalised in order to be reused. When abstraction is viewed as a way of normalising knowledge then abstraction seems a logical requirement of reuse.

Clancey sees the key to enabling reuse is the use of abstract terms for the description of the domain and control knowledge. To generalise a domain specific rule it is necessary to replace specific terms with a more abstract term and then to add in the relationships. New relations and the reclassification of expressions may be necessary to enable reuse. He advocates the inclusion of metarules to allow the knowledge to be processed more flexibly by reordering clauses, not just reordering rules. Metarules need to be general enough to be reusable and specific enough to enable reuse, that is they need to be comprehensible.

Clancey (1992) notes that a different type of reuse occurs when abstraction of control or domain knowledge is used more than once in a program for different circumstances. He states "the knowledge base is easier to construct because the expert need not specify every situation in which a given fact or relation should be used" (Clancey, 1992, p.17) The knowledge base is therefore less brittle and can use the facts and relations for different purposes. If a new kind of knowledge is to be added then new metarules and relations may be needed.

Research by Richards (1994) also found that through the use of metarules, in the form of WHEN-CHANGED *methods,* it was possible to adapt a consultation style ES to behave like a decision support system (DSS) by supporting 'what-if' analysis. In that study no changes were made to the domain knowledge only to the way the domain knowledge was used.

Research on knowledge of the visual process has found that there is "a strong correspondence between knowledge organisation and knowledge use" (Tsotsos 1982). The aspects of context and the abstraction of form and motion knowledge into an hierarchy are considered important to adequately represent visual knowledge. Many describe the need for abstraction without giving many clues to what levels may exist. The following list by Tsotsos (1982) of types of concepts that may need to be represented could be used to identify categories:

- prototypical concepts and instances
- discrete and structured concepts
- ordinality and measure
- properties, qualities and attributes
- causality

- spatial knowledge
- temporal knowledge
- states, events, actions and change
- procedural knowledge
- situations and contexts
- description by comparison and differentiation
- conjunction, disjunction and negation
- inheritance, instantiation and reasoning with defaults
- certainty, strength of belief
- expectations

These concepts could then be organised by:

- plurality (sets, sequences, membership, partial orders)
- projection
- abstraction
- multiple viewpoints
- or meta-knowledge.

The use of levels of abstraction has been particularly advocated for the purposes of explanation. Chandrasekaran, Tanner and Josesphson (1988) argue that most knowledge representation methods are at too low levels of abstraction to provide adequate explanation. Swartout and Moore (1993) categorise explanation approaches into four main groups, three of which use abstraction for explanation. These first two approaches have been considered already and are: the use of metarules and the use of generic problem solving methods such as generic tasks. The third group is Swartout and Johnson's work on Explainable ES (EES) that use an automatic programmer to provide varying levels of knowledge and avoids the loss of what he terms *compiled out* knowledge. Each group uses different types and numbers of levels but they all tend to have levels that range from concrete to purely abstract, or from specific to general. The absence of different levels of abstracted knowledge has been a shortcoming of RDR which this thesis has addressed. This work is described in Chapter Five where FCA has been used to find the intermediate and higher levels. A more detailed discussion is given in Section 7.3.3 where the explanation activity is considered. An issue similar to abstraction is the concept of deeper knowledge. We look at this next.

2.2.3 The Need for Deeper Knowledge

A number of different terms, such as deep versus shallow, derived versus direct and deep versus compiled, are used by different researchers to express the notion of levels of knowledge. Chandrasekaran and Johnson (1993) use the terms direct and derived knowledge. As the terms imply, direct knowledge is readily available but derived knowledge requires computation, perhaps via a subtask, from existing knowledge or it must be acquired from the external environment. Derived knowledge can be seen as a *deeper* level because it does not exist on the surface but requires moving deeper into the knowledge performing various procedures until it can be generated. Chandrasekaran and Johnson (1993) explain that depth is a relative notion and the depth of one knowledge element to another can be given in terms of the amount of processing needed to get to each of the elements.

Further understanding is gained by looking at some other related distinctions. Swartout and Moore (1993) speak of deep versus compiled knowledge. Chandrasekaran and Johnson (1993) equate direct knowledge with compiled knowledge because it can be used without the need for further processing. The deep knowledge is the knowledge that is given as a prerequisite for system development. For example, it may be knowledge about the domain, organisation, work and data-flows, various computer files and documents, recorded discussions and so on. Different parts of the available knowledge are used to develop the appropriate system. Often only what is needed to solve the predefined task is kept. Much of the other knowledge is conceptually lost or *compiled-out* and can not be utilised by the system. This is seen as a major cause for the brittleness of ES and their inability to provide adequate explanations.

Another analogy is the comparison between model-based and rule-based reasoning (Chandrasekaran and Johnson 1993). They see the model as being an example of deeper knowledge where more general knowledge is represented than in the case of a specific rule. The model contains “the principles of the domain while rules are supposed to refer to relatively *ad hoc* associations between evidence and hypotheses” (Chandrasekaran and Johnson 1993, p.250).

Finin and Klein (1989) describe the pros and cons of *deep* and *shallow* models. Shallow models are generally easier to build because the knowledge is encoded in the heuristics where the conclusion is drawn directly from the facts. This avoids much of the complexity of trying to get experts to define how they arrived at a conclusion. This also means that inferencing is straightforward and efficient. However, shallow models contribute to the "brittleness" problem of ES. Since the knowledge is not explicit, they are inflexible, can be difficult to maintain or modify and are inadequate for explanation. Deeper models contain explicit knowledge in a manner that can be equated to reasoning from first principles. Finin and Klein (1989) see them as more robust and more verifiable for completeness and correctness. Deeper models tend to handle new problems better and facilitate better explanation. As a trade off, the reasoning process is more complex and slower.

As Finin and Klein point out, there is much discussion on the development of deeper models but there is little guidance on how to build these deeper models. They propose:

1) using an informal definition of "knowledge depth" using the *deeper-than* relation that is applied to sub-tasks,
2) abstracting of high-level reasoning tasks such as diagnosis or simulation and
3) analysis of the relative depths of reasoning task models found in various domains.

Kidd (1985) offers an alternative structuring system based on functional requirements of remedies, which may be broken into three levels: symptom, disorder and cause. The approach is aimed at man-machine symbiosis, but is limited to diagnostic systems only.

The type of knowledge contained in an RDR KBS is hard to define using these various and sometime conflicting terms of deep, shallow, direct, derived and compiled. The assertional RDR KBS can be seen as the direct (Chandrasekaran and Johnson 1993) knowledge. And the terminological KBS that FCA produces can be seen as the deeper derived knowledge. If we use Swartout and Moore’s (1993) terms, the RDR KBS is more like deep knowledge which includes the cases and the behavioural knowledge. Neither the RDR assertional KBS or the terminological KBS that FCA develops fit with notion of compiled knowledge since in both cases no knowledge is lost but with FCA more knowledge is generated in the form of higher concepts. Even though the line diagram does not make use of the cases, it is shown in Chapter Six how the line diagram can be used to pop-up the case associated with a particular concept. If we use Finin and Klein’s (1989) definitions, an RDR KBS is made up of shallow knowledge that is contained in the heuristics where conclusions are drawn directly from facts. The RDR knowledge does offer the benefit of ease of capture but does not suffer from the maintenance problems associated with shallow models. The extent to which RDR suffer from the brittleness problem is unclear however the ability to support explanation has been seen as an area that needed improvement in RDR. This study acknowledges that knowledge of different types and depths is needed to support *activity-reuse* and is addressed in the incorporation of FCA into RDR described in Chapter Five. We look now at the last issue which is the need for more knowledge.

2.2.4 The Need for More Knowledge

Another finding of the various reuse studies is that extra knowledge may be needed when the usage of the knowledge changes such as using knowledge intended for a consultation system for causal modelling (Lee and Compton 1995) or for explanation (Buchanan and Shortliffe 1984).

Clancey (1992) also concludes that although the SSM perspective is useful for a range of problem types, knowledge that is needed to solve one type of problem may not be adequate to solve another type of problem. It is possible to achieve something without referring to what is conceptually being done. This means that it becomes difficult when we take knowledge out of context and try to apply it to a different situation and that contextual knowledge is one type of knowledge that may be need to be added.

The study by Richards (1994) found that extra knowledge in the form of rules was not necessary when adapting to a 'what-if' system style, although additional models to handle the aspect of time had been added. It seems reasonable that different situations may impose different knowledge requirements and the question of whether more knowledge is needed can be best summed up by Hemmann:

> *"Of course, at least the domain model must either contain redundancy to be combinable, or it must be extended by the knowledge engineer according to the knowledge needs of the individual interpretation models" (Hemman 1992, p. 6).*

In this thesis the capture of additional knowledge has not been necessary but the derivation of additional knowledge in the form of higher level concepts and the knowledge structure has been found to be necessary to support *activity-reuse*. In the next section we summarise the findings of this chapter and the main issues of relevance to *activity-reuse*.

2.3 The Issues Relevant to Activity-Reuse

This section only briefly mentions the main issues identified above concerning contextual, abstracted, deeper and more knowledge as they relate to *activity-reuse* since they are covered in other sections of this thesis. Capturing knowledge in context is a key benefit offered by RDR and is discussed in Section 3.2.2. The need for abstracted knowledge became apparent when looking at various reflective modes to enable the higher level concepts in the domain to be found. The work is reported in Chapter Five which looks at building a subsumption hierarchy from the primitive concepts in the RDR rule-base. This abstracted knowledge fits Chandrasekaran and Johnsons (1993) definition of deep knowledge as being model-based knowledge. The decision-list structure of an MCRDR KBS can also be used to provide layers in the explanation path because the higher level rules offer more general principles and the refinements describe the exceptions to the general rule (Catlett 1992). Swartout and Moore's definition of deep knowledge corresponds more to background or supporting knowledge and to some extent is satisfied by the use and storage of cases in RDR which are used both for KA and for explanation purposes. With respect to the need for more knowledge this study has not found it necessary to capture more knowledge but the derivation of higher level knowledge using formal concept analysis can be seen as a form of learning more about the domain. Some activities, such as causal modelling, may require additional knowledge. Lee and Compton (1985) have looked at causal modelling using RDR KBS and have hypothesized that much of the additional information needed can be acquired as a rule is added with minimal increase in difficulty or work load on the user's part.

Many of the reuse approaches presented attempt to develop generic approaches which could be applied to other situations. As enumerated by Rademakers and Vanwelkenhuysen (1993) in Section 2.1.3. there are serious problems with the use of generic models, the key problems being matching the best PSM with the problem and then finding the best way to adapt the method. There is also then the problem of how to handle linking of the PSM and the domain model (Hemmann and Voss 1993a). To avoid these issues what is really needed is a representation that can handle the problem regardless of the problem categories into which it may fall. RDR research is heading in this direction as shown in the recent work which required a small modification to the MCRDR inference engine to allow the one problem solving method to be used for classification and configuration tasks (Ramadan et al 1998) and a different alteration to support a control problem (Shiraz and Sammut 1998).

As explained, this study is looking at reuse from a different angle and is primarily concerned with the reuse of knowledge within the domain for which it was captured to suit a range of user needs. The focus on PSMs, ontologies and KBS components is often concerned with reuse across domains. User issues are of lesser importance to these approaches than seeing if something that has worked in one domain is transportable to another.

Activity-reuse will involve investigating questions such as “What are the constraints on the usability of knowledge recorded for different purposes ?” (Neches et al 1991, p.49). This question was asked by the KSE group but does not appear to have been answered by their endeavours. The research reported in this thesis is concerned with this question. In Chapter Seven a description of each activity that was explored and their particular requirements are given.

There are two aspects that appear relevant in pursing a solution to reusing knowledge for different activities. One aspect is the reuse of knowledge already captured and the other is building KBS designed for *activity-reuse*. The first approach looks at taking an existing KBS and reusing the knowledge therein. This will involve what Hemmann (1993a, p.2) loosely defines as "careful analysis" and he cites Clancey's work on

MYCIN in developing the idea of Heuristic Classification as an example of such a technique. It is interesting that despite the strong emphasis Hemmann gives to the use of formal methods such as KADS and the even more formal methodology he has built called MoMo, he offers very little concrete guidance on how to analyse what already exists. Descriptions such as "concentrated on the code" (Hemmann 1993a, p.3) are not very helpful. This study is interested in a stronger solution to reusing existing knowledge than "careful analysis" and the use of rough set theory (Pawlak 1982) and formal concept analysis (Wille 1982) are two techniques that were investigated for this purpose. Chapter Four and Five describe these theories and the work done with them.

The second approach is really the most important. An outcome of this thesis is a general solution to building KBS that are reusable for multiple activities. It seems logical that something that is built from the start with reuse in mind will be better than a system that needs to modified later to cater for reusability. The interesting thing with RDR is that this has not been the experience of this study. The simplicity and the flexibility of the approach meant that using an existing system did not constrain the development of a reusable system. The approach provided offers a way of capturing assertional knowledge and deriving terminological knowledge that is then considered adequate to support a range of activities with the burden on the user interface to support these different uses.

The discussion in Section 2.1.3 covering the limitations of the knowledge level and generic solutions is also important because the approach adopted in this study does not follow these approaches but rather emphasizes solutions that do not rely on the user to state their own conceptual models. The prerequisite that knowledge level modelling must be performed before KA can commence is a major weakness of these approaches since they add to the KA bottleneck and do not acknowledge that although a knowledge level may exist we do not really understand it or how the human mind works. It also appears that if humans use a knowledge level they tend to use it after the event as a means of reflection. An alternative approach is offered by RDR which simply and easily captures behavioural knowledge that can be validated in conjunction with the cases used in the KA process. If has been found during the course of this study that behavioural knowledge is not enough and that some understanding of the conceptual or mental model is necessary. FCA has allowed the derivation of these higher level models from the performance knowledge and it is argued that a post-hoc approach to the development of such models is a feasible and possibly more reliable or verifiable approach.

Up till now a number of claims, but little detail, regarding RDR has been given. This omission will be remedied in the next chapter which describes RDR, its variations, implementations, strengths and weaknesses. In addition other related theory is discussed, namely personal construct psychology and case-based reasoning. We also look at a nearest neighbour algorithm that was developed to determine the closeness of concepts in the KBS and what limitations of RDR this thesis has addressed in the pursuit of a multi-purpose system.