CHAPTER 1: INTRODUCTION

1.1 Chapter Outline

This chapter provides the context for the research together with the objectives, motivation, research questions, significance, outcomes, methodology, and thesis organisation.

Please note that a glossary is provided at the end of this thesis (page 277) to assist with the acronyms and jargon used.

1.2 Context

For many different types of data, and in many different domains a simple keyword search using popular search engine technology can produce volumes of false positive hits. The ordering of search hits may be based on metrics such as statistical click-through rates, or anchor text[5] that is inappropriate or more troublesome still, context-sensitive leading to false positives and false negatives. According to one recent study (Gully and Signorini, January 2005)[6], there are over 11.5 billion web pages in the publically-indexable Internet. Given the size and complexity of some domains popular search engines provide a shoe that simply can't fit all.

The problem of too many false-positives is exacerbated when users try to use simple keyword search algorithms to search numeric data. Popular search engines lend themselves to straight keyword searching but at this time they are unable to effectively locate and prioritise relevant answers in problem spaces dominated by complex functionally derived, context dependent and / or numeric attributes[7]. Search appliances typically do not close-the-loop on system searches and hence build no expertise in the accuracy of context sensitive query hits. Their ability to learn the relevance of search results in relation to the search context is minimal.

---

[5] For a definition of Anchor Text, please see the glossary (page 277).

[6] http://www.cs.uiowa.edu/~asignori/web-size/

[7] For example, if you search for "3" in Google, the search results give you dates including the number 3, companies whose names include the number 3, promotional material including the number 3, but nothing about the number 3 itself. The context in which you were searching for the number 3 is obviously vital in determining the most appropriate search result.

Similarly, data mining techniques allow question-answer indexes to be developed in mature domains where unambiguous domain models have already been constructed, and a large dataset of pre-classified question-answer pairs are available. But for previously unchartered real-time functionally complex, repetitive and evolving problem domains, static and off-the-shelf computer-driven search and data mining techniques demand significant and often prohibitive levels of customisation and manual refinement (Dunham, 2003).

For these types of problem domains, manual, customised and ongoing knowledge acquisition from real humans of the attributes of the domain (the *domain model*), the conditions that map cases to classifications in the domain (the *matching criteria*), together with the classifications and their associated conclusions, is needed. An approach is needed that can be customised to the domain and data at hand to provide much richer context specific pattern matching.

The knowledge acquisition techniques of Single Classification Ripple Down Rules (SCRDR) (Compton and Jansen, 1989) and Multiple Classification Ripple Down Rules (MCRDR) (Kang, Compton and Preston, 1995) (described in detail in Chapter 4 on page 40) have previously been applied with reported success as single-user result-interpretation indexing techniques in numerous domains[8], and perhaps most notably in the pathology domain (Edwards, 1996) (Compton et al. 2005).

MCRDR has also been previously applied to the help-desk domain, but only as a single-user keyword-based index for web-based documents (Kim, 2003) (Kang et al., 1996). (Further description of these implementations is provided in section 4.3.2.3 on page 58.)

From this experience, it was conjectured that the SCRDR and MCRDR techniques might be useful in improving the effectiveness and efficiency of problem-solution matching and hence trouble-shooting in support centres operating in previously unchartered real-time functionally complex repetitive and evolving problem domains.

The last 15 years have seen globally explosive growth in support-centres providing help and service-desk functions: advising customers, answering queries, and resolving customer problems. A primary objective for support-centres is to learn how to solve technical problems with minimal heartache (for the sake of the customer), headache (for the sake of the staff),

[8] Ripple Down Rules (RDR) has been studied in numerous domains, including Pathology, Email sorting, Flight Control, Chess, Traffic Light Control, VLSI Channel Routing, Help-desk, and RoboCup.

and resources (for the sake of the business). The ability to find relevant solutions in the Information and Communications Technology (ICT) support domain is notoriously difficult due amongst other things to the lack of anchor text, and the presence of numeric data (e.g. hexadecimal error codes) and industry specific jargon (such as version codes) that resists plain text keyword searching.

The support centre problem domain would offer new challenges for MCRDR compared to some of the previously studied domains: evolving cases, evolving solutions, multiple contributors to the evolving knowledge, and a need for real-time responsiveness in the knowledge based system (KBS). The Sydney-based support centre of a 24x7 follow-the-sun[9] global support operation of HTG, a large multi-national corporation operating in the ICT industry therefore provided a suitable environment in which to conduct the research.

HTG has sales channels in more than 50 countries and annual revenues of ~$US 10 billion. It is one of the world's 10 largest ICT companies as measured by market capitalisation, and it employs more than 25,000 staff globally. The trouble-shooting environment at HTG was characterised by approximately 5000 incoming problem cases per 24 hour period around the globe. Around 1000 of these were cryptic and repetitive computer-encoded cases. Much of the trouble-shooting know-how was stored in people's heads, which was problematic since the organisation suffered ongoing staff turnover. As well, written information required to solve the incoming problems was scattered across numerous and diverse locations.

1.3 Objectives and Motivation

The aim of the research was to:

> *Develop a knowledge acquisition approach that will facilitate the capture and recall of problem solving know-how, in an environment where problems and their solutions are continuously evolving, and where a distributed group of stakeholders can both contribute to and benefit from the acquired knowledge.*

[9] "24x7 follow-the-sun" is an industry term that refers to an alliance of support centres around the globe handing over to each-other e.g. every 6-8 hours so that employees can work in daylight hours while customers can be supported around the clock. For example, after 6 hours of operation a support centre in Sydney hands over to India, that hands over to Ireland 6 hours later, that hands over to East Coast USA after another 6 hours, that hands over to Sydney after a further 6 hours, after which the handover cycle repeats itself.

This aim was motivated by a desire to improve the quality and consistency of solutions together with the speed of solution recall, and hence the effectiveness and efficiency of trouble-shooting.

A sub-goal of this research was to explore the possibilities to adapt SCRDR and / or MCRDR to the support centre context, and to design, prototype and test a software *blueprint* that could assist with problem solving in previously unchartered, repetitious and complex problem domains.

A further sub-goal was to develop an approach that would be flexible enough to support collaborative trouble-shooting and classification in other domains such as botany, zoology, biology, chemistry, pathology, geology, and financial markets.

1.4 Research Questions

The questions answered by this research[10] are shown in Table 1. The location of the corresponding research answers is also shown.

[10] These research questions were constructed retrospectively to improve the presentation of this thesis.

Table 1: Research Questions

Research Question	Answer Location
Q1. What does the HTG support centre environment tell us about the nature of trouble-shooting?	Chapter 2, page 13.
Q2. What are some of the problems with existing knowledge management approaches in domains such as the ICT support centre?	Chapter 3, page 30.
Q3. What literature and technologies are relevant to a collaborative MCRDR-based troubleshooting approach?	Chapter 5, page 64.
Q4. How can MCRDR be adapted to a real-time environment where problems and their solutions are continuously evolving, and where a distributed group of stakeholders can both contribute to and benefit from the acquired knowledge?	Introduction - Chapter 6, page 79. Top Level Design - Chapter 9, page 161. Detailed Design - Chapter 11, page 176. Proposed Design Enhancements - Chapter 13, page 242.
Q5. What is the expected trajectory of the case-driven acquisition of classification knowledge?	Chapter 7, page 92.
Q6. Why take a hybrid case-based and rule-based approach to Knowledge Acquisition?	Chapter 8, page 133.

1.5 Significance

As discussed later on in section 5.2 (page 66), the *cooperative* work offered by Richards (1998) and (Beydoun et. al, March 2005) and (Beydoun et. al 2007) takes separately created KBSs from independently operating experts modelling identical domains and merges them, for example on a weekly or daily basis. In contrast, the *collaborative* work offered in this thesis allows experts to concurrently negotiate and build an expert system together, while being informed of each-others present viewpoints. This research is significant because it extends the previous Ripple Down Rules (RDR) techniques to offer a novel knowledge representation and Knowledge Acquisition (KA) algorithm that supports both the bottom-up

case-driven and top-down[11] rule-driven concurrent *collaborative* acquisition of the domain model and case-classification matching criteria. For the first time, collaborative MCRDR (C-MCRDR[12]) is offered. In this research, a C-MCRDR prototype system was proposed, designed, prototyped and tested.

As well, a mathematical model has been developed (Chapter 7, page 92) that allows predictions to be made for the stochastically expected knowledge acquisition trajectories of case-driven classification systems. The derived case-driven KA model is relevant to the case-driven acquisition of classification knowledge for example in Artificial Intelligence, Machine Learning, Data Mining, Expert Systems, Ripple Down Rules, Group Decision Support Systems, Collaborative Tagging, Folksomonies[13], and Case-Based Reasoning (CBR) systems. The case-driven KA model offers important predictions for the trajectories presented in previous machine-learnt and case-based KA simulations for SCRDR and MCRDR as discussed in (Compton, Preston and Kang, 1995), (Kang, Lee, Kim, Preston and Compton, 1998), and (Cao and Compton, 2005 and 2006) as well as the tag-acquisition trajectories discovered by (Goldman and Huberman, 2005, p4) for folksomonies. Further related work that may be informed by this research is discussed in section 7.1 (page 92).

[11] In this thesis, the term "top-down" knowledge acquisition refers to the process of human users adding rules directly to a rule tree, without needing to refer to cases. The term "bottom-up" knowledge acquisition refers to human users adding rules on the basis of a specific case on hand, in a manner that does not necessitate the user to peruse the rule tree, for example by allowing the user to create new RuleNodes that are relative to existing RuleNodes in the system. Please see the Glossary (page 277) for a definition of the term RuleNode.

[12] The collaboration techniques proposed in this thesis apply equally to SCRDR or any of the other RDR alternatives, for example those mentioned later on in section 4.4 (page 60). Hence the terms C-SCRDR, C-RDR and C-NRDR are also proposed by this research.

[13] Collaborative Tagging systems and Folksomonies are described in section 5.4 (page 72).

1.6 Outcomes

As mentioned in the synopsis for this thesis, outcomes from this research include:

1. A review of the trouble-shooting environment at HTG, showing that trouble-shooting know-how includes the ability to configure (i.e. work-up) a case, classify it, and locate its relevant solution.
2. The derivation of a stochastic model that explains and provides predictive formulas for Case-driven Knowledge Acquisition as in Single Classification Ripple Down Rules (SCRDR) systems[14], Multiple Classification Ripple Down Rules (MCRDR) systems[15], and Collaborative Tagging Systems such as Folksomonies[16].
3. A knowledge representation and knowledge acquisition technique known as 7Cs that supports the **C**ollaborative **C**onfiguration and **C**lassification of a stream of incoming problem **C**ases via a set of **C**onditionNodes linked to their **C**lasses and associated **C**onclusions.

In the author's opinion, the most important of these three main outcomes is outcome 2: the stochastic case-driven KA model presented in Chapter 7 (page 92).

1.7 Methodology

The following seven research instruments have been used to direct this study: literature review, case study, interview, survey, mathematical derivation, software prototyping, and software user-trial. The support centre offered a suitable case study in which the more general problems associated with collaborative trouble-shooting in a dynamic problem domain could be studied.

Numerous interviews were conducted together with a major survey of the support centre to determine the nature of the trouble-shooting organisation, tasks, processes, and workflow (the

[14] (Compton and Jansen, 1989).

[15] (Kang, Compton and Preston, 1995)

[16] For a discussion of Collaborative Tagging systems and Folksomonies see section 5.4 (page 72) and Vander Wal (2005) .

ethics approval reference number at Macquarie University is HE25FEB2005-D03895). The results of the interviews and survey are reported in Chapter 2 and Appendix A.

A technology and literature review was conducted[17]. The results of the literature review are reported in Chapters 3, 4, and 5.

An interview was also arranged with the chief technology officer at Pacific Knowledge Systems (PKS) to learn from their experiences in deploying the Multiple Classification Ripple Down Rules (MCRDR) technology in the pathology domain. Results of this interview are included in Appendix G.

A predictive stochastic model was developed for the rule-driven versus case-driven transfer of classification knowledge. This model is presented in Chapter 7 and some of the implications are discussed in Chapter 8.

A large part of the research effort was spent in developing prototype software for the support centre, and following a recursive Plan-Do-Check-Act (PDCA) improvement cycle (Deming, 1989) to make the best possible fit between the needs of the support centre and my software solution. A final patent application was the net result of these efforts (U.S. Patent application PCT/AU2005/001087). The top-level design, problem context, detailed design, implementation, trial and validation of the prototype are reported in Chapters 9 though to 12.

The methodology applied has some parallels with the CRISP-DM[18] (Cross Industry Standard Process for Data Mining) KDD (Knowledge Discovery in Databases) model. CRISP-DM contains the following steps: business understanding, data understanding, data preparation, modelling, and evaluation deployment (Dunham, 2003 p19). The steps involved can be summarised as the 5As: assess, access, analyse, act, and automate.

1.8 Thesis Organisation

The remainder of this thesis is organised as follows[19]:

[17] A review of vendor offerings for the support centre was also conducted, but it is not reported on in this thesis.

[18] www.crisp-dm.org

[19] For the executive reader and in the author's opinion, the most relevant chapter to the thesis topic is Chapter 5, the most interesting is Chapter 7, the most controversial is Chapter 8, the most enlightening is Chapter 11, and the most promising is Chapter 12.

Chapter 2: The HTG Support Centre Environment

Documents the findings of the observations, interviews and surveys conducted at the support centre.

Chapter 3: Other KM Approaches

Provides a review of other KM approaches relevant to the main research question.

Chapter 4: Ripple Down Rules

Provides a detailed review of the RDR literature.

Chapter 5: Collaboration Features and Trends

Argues a case for including support for collaboration in the RDR approach.

Chapter 6: Adapting MCRDR

Describes the limitations of the conventional MCRDR framework that surface when we migrate the algorithm to the support centre domain.

Chapter 7: A Model of Knowledge Transfer

Provides a predictive model for case-driven and rule-driven knowledge transfer.

Chapter 8: Hybrid Case and Rule-driven KA

Discusses the design implications of the model derived in the previous chapter.

Chapter 9: 7Cs Top Level Design

Documents the top-level design of the proposed 7Cs solution for the support centre, and for trouble-shooting more generally.

Chapter 10: The Dial-Home Problem Context

Provides a description of the target problem domain for the *FastFIX* software trial. Please note that FastFIX is the name of the prototype software developed in this thesis in order to trial the 7Cs design concept.

Chapter 11: The 7Cs and FastFIX Design Core

Documents the detailed design of the 7Cs prototype system and presents the implemented FastFIX software prototype.

Chapter 12: Results of the FastFIX Trial

Analyses the results of the FastFIX software trial in light of previous RDR evaluations and the model presented.

Chapter 13: Design Enhancements

Documents ways in which this design might be enhanced in the future.

Chapter 14: Summary and Conclusions

Provides a summary and conclusions for this research.

Other Sections

A glossary is provided to allow easy-lookup for any acronyms and jargon used herein. This is followed by a list of publications resulting from this research, details of the patent filed, details of the innovation award received, and a reference list. Finally, the following Appendices are provided:

Appendix A – Trouble-shooting Survey

The complete results of the HTG trouble-shooting survey are provided.

Appendix B – Sample CaseDB[20] Case

Demonstrates a sample problem case from HTG.

Appendix C – System Requirements

Documents the broader system requirements for the proposed trouble-shooting system.

Appendix D – Concepts from the Literature

Includes key concepts from the Artificial Intelligence and Knowledge Acquisition fields.

Appendix E – Tacit Knowledge

Provides interpretations of the term "tacit knowledge" relevant to this research.

[20] CaseDB is an alias for HTG's case tracking software.

Appendix F – Ontologies

Provides a discussion of the support for collaboration in present day Ontology editors.

Appendix G – Transcript of PKS Interview

Details the interview arranged with PKS to discuss the experiences they had implementing MCRDR in the pathology domain.

Appendix H – About Anchor Text

Describes the importance of anchor text to popular search engines as reported by CSIRO's Panoptic search engine project.

Appendix I – Folksomonies

Reviews two examples of folksomonies: del.icio.us and flickr.com.

Appendix J – Wikipedia versus Britannica

Provides a summary of Wikipedia's rise and dominance over Encyclopaedia Britannica in the last decade.

Appendix K – Semantic Web Languages

Summarises some of the languages promoted by present day semantic web approaches.

Appendix L – DARPA Knowledge Sharing

Provides a brief literature review of the DARPA Knowledge Sharing Effort.

Appendix M – The FastFIX Prototype Shell

Documents the application framework within which the 7Cs prototype system was built.

Appendix N – Acquired RuleNodes

Provides a list of RuleNodes acquired during the FastFIX software trial.

Appendix O – 7Cs Implementation Enhancements

Documents some of the implementation enhancements recommended for future embodiments of the 7Cs system.

Appendix P – Limitations of MCRDR – An Example

Presents a KBS example that illustrates a number of the limitations of conventional MCRDR.

Appendix Q – Rough Knowledge

Discusses the process of knowledge transfer where some of the knowledge is false, speculative or unknown.

Appendix R – RuleNode Relationships

Discusses some of the RuleNode relationships that could be managed in a future case- and rule- driven (C.A.R.D) 7Cs system.

1.9 Chapter Summary

This chapter has introduced the context for the research together with the objectives, motivation, research questions, significance, outcomes, methodology, and thesis organisation.

The next chapter provides a review of the support centre environment in which the research was based.

CHAPTER 2: THE HTG SUPPORT CENTRE ENVIRONMENT

2.1 Chapter Outline

This chapter addresses the following research question:

> *Q1. What does the HTG support centre environment tell us about the nature of trouble-shooting?*

The findings of the interviews, observations and a survey conducted at the HTG Sydney-based support centre are presented. The findings highlight the knowledge management issues faced by trouble-shooters, the nature of trouble-shooting in general, and the design constraints that a new trouble-shooting information systems approach would need to satisfy within the context of HTG.

Recalling from section 1.3 on page 3 that the aim of the research was to:

> *develop a knowledge acquisition approach that will facilitate the capture and recall of problem solving know-how, in an environment where problems and their solutions are continuously evolving, and where a distributed group of stakeholders can both contribute to and benefit from the acquired knowledge;*

for the purpose of this thesis, the HTG support centre provides a case study that offers insights to the problem solving (i.e. trouble-shooting) process in general. While sections C.3 to C.10 (commencing on page 392) in Appendix C discuss some of the relevant support centre literature, the focus of this thesis was not to survey support centres. Rather, the focus was to help trouble-shooters in any problem domain quickly and accurately locate solutions.

In summary, as a result of the interviews, observations and survey presented in this chapter, it was found that trouble-shooting includes the ability to configure (i.e. work-up) a case, classify it, and locate its relevant solution. Problem solving appeared to follow a case-configuration-classification-conclusion cycle[21]. As well, multiple classifications could apply to a problem,

[21] Problem solving as a classification and configuration problem has previously been discussed in (Clancey, 1984) and (Clancey, 1985). Richards (1998a, p26) discusses types of problems, as does Breuker (1994). Breuker's eight problem types fall into two broad categories which he calls *synthesis* and *analysis*. These are

and several different solutions may need to be tried. It was found that problem solving at the HTG support centre required diverse types and sources of knowledge, and that much of the case-configuration-classification-conclusion knowledge was not shared, but rather stored in the minds of individual experts.

2.2 Business Context

As mentioned previously (section 1.2, page 1), HTG's support centre experiences upward of 5,000 customer problem cases per day globally. 20% of these cases arrive in the form of cryptic error codes in hexadecimal format automatically emailed from errant equipment to the support centre's case tracking software in the form of "dial-homes". The nature of HTG's dial-home problem cases is described in more detail in Chapter 10 (commencing on page 169) and an example problem case is provided in Appendix B (commencing on page 387).

At HTG, non dial-home cases (4,000 per day) each take on average 2 hours to solve. The estimated direct labour cost of solving the non dial-home problems is more than $105 million per annum. In contrast, dial-home cases (1,000 per day) each take on average 15 mins to solve. The estimated direct labour cost of automatically solving the dial-home problems is approximately $3.3 million per annum. In fact the whole support operation includes its own support staff, infrastructure and management - approximately 4,800 staff, and HTG's cost of services as a whole are the order of $US1 billion per annum (2004 Annual Report).

Obviously even a small percentage saving in the labour required by the support centre would have a significant impact on HTG's profitability, not to mention the flow on benefits for customers.

With better information about the types of problems being experienced by the customer, HTG could also strengthen the performance and reputation of its products and services.

2.3 Health Check Interviews

To better appreciate the issues facing knowledge-workers at the HTG support centre, interviews were conducted with experienced trouble-shooters and management at HTG, and

---

more commonly known as *construction* and *classification*. Configuration is one particular type of construction problem, for example the Sisyphus elevator problem (see http://ksi.cpsc.ucalgary.ca/KAW/Sisyphus/).

the operational issues facing them were discussed. The day-to-day activities of support staff were also observed and some of the training available to new hires was undertaken. The outcome of the interviews, observations, training sessions and active participation was a (commercial in confidence) "Health-Check" report that I wrote (24th July, 2003) to document the current situation. This report was reviewed by HTG and formed the basis of the 7Cs and FastFIX[22] system design goals.

2.3.1 Types of Knowledge

During the health check interviews, it was found that trouble-shooting personnel were relying on at least four disparate sources of (explicit) knowledge when solving problems[23]:

- *Engineering Knowledge*: How does the product work?
- *Operational Knowledge*: How do you use it?
- *Interoperability Knowledge*: How does the product interact with third party products?
- *Problem Solving Knowledge*: How do you fix it?

As well, it was found that the trouble-shooting process required personnel to use a great deal of unspoken (tacit[24]) problem solving knowledge, including:

- *Problem Determination* Knowledge i.e. what is the class of problem on-hand?
- *Search Location* Knowledge i.e. where should I search for a solution?
- and the *Search Criteria* to be applied i.e. what parameters should I use in my search for a solution?

It appeared that the capture of this explicit and, perhaps more importantly, tacit problem-solving knowledge was missing from case and solution tracking vendor software.

---

[22] As mentioned in section 1.8 (page 8) and in the glossary (page 277), FastFIX is the name of the prototype software developed in this thesis in order to trial the 7Cs design concept. The 7Cs and FastFIX top level design are presented in Chapter 9 (page 161), and the FastFIX detailed design is presented in Chapter 11 (page 176).

[23] Of some relevance is the problem types identified by Hayes-Roth et. al (1983) referred to by Clancey (1985, p29).

[24] For a discussion of tacit knowledge, see Gourlay's (2002) references to Polanyi (1958), Nonaka and Taceuchi (1995) and Appendix E (page 404).

2.3.2 Complexity and Changeability

During the health check interviews, it was found that the following factors appeared to contribute to the complexity of problems coming into the support centre[25]:

- *Technology Convergence* – many technologies coming together into integrated multi-function solutions.
- *Vendor Divergence* – many vendors providing solutions in the same technical space but with different and sometimes conflicting interface requirements.
- *Product Evolution* – iterative software and hardware product releases leading to multiple layers of potential problems and solutions.
- *Knowledge Evolution* – greater human understanding of the problem domain with the passage of time.
- *Conflicting Expert Opinion* – conflicting views on how to solve a wide range of problems possibly as a result of there being no one person with expertise across all the necessary domains.

These factors were creating a complex and dynamic knowledge milieu, and a seemingly endless search space for solutions.

On the upside however, a large percentage of the problems appeared to be repetitive, even if only in terms of the background knowledge required to solve that particular class of problem, allowing for the possibility of solution re-use. The repetition rate for problem cases at HTG is discussed further in section 2.4 (commencing on page 17).

2.3.3 Sources of Knowledge

It was also found that:

- Much of the knowledge was stored in people's heads rather than explicitly documented in technical references.

[25] Some of this text appears in (Vazey & Richards, 2005b).

- Existing documentation often missed the necessary detail or was ambiguous, and a significant amount of technical product information was cryptic at best, coming in the form of abbreviated slides, videos, or emails.
- Personal relationships were extensively and often exclusively relied upon to source basic product information from fragmented silos of knowledge scattered throughout the company.

2.3.4 Employee, Customer and Business Impact

In the worst case, the impact of these fragmented knowledge networks appeared to be a poor level of knowledge re-use resulting in:

- Increased frustration;
- Duplication of effort;
- Slower problem resolution;
- Inconsistent customer responses;
- Customer dissatisfaction; and
- Overall organisational inefficiency.

High staff turnover rates were both an outcome and a contributing factor.

2.4 Observations

2.4.1 Incumbent Knowledge Systems – CaseDB and SolutionDB

In the last 10-20 years, information systems at support centres have been dominated by *Case Tracking* solutions, recording case information such as: Who raised the call? What is the product in question? What operating system is being used? Which engineer is the problem assigned to? etc etc. The purpose of such *Case Tracking* solutions is to allow management and staff to track the progress of customer problems as they pass through the customer service organisation. At the time of this research[26], the HTG support organisation was using

[26] Research at the HTG support centre was conducted from July 2003 to December 2005. Thesis write-up was commenced in December 2005 and a complete draft thesis was submitted for review in May 2006. Submission of the final thesis was delayed by several factors including protracted negotiations to protect the commercial

CaseDB[27] as its case tracking system. An assessment of CaseDB's fitness for purpose is provided in the responses to survey questions in Appendix A (commencing on page 352, with specific comments commencing on page 354).

At HTG cases could be machine generated e.g. with problem equipment ringing or emailing through the problem tickets. They could also be entered directly by customers, for example via a user-driven web interface. The more traditional scenario was where the problem case originated with a front-line customer service officer taking the first call.

As is typical at support centres, at HTG the problem case / ticket passes through several states in which it moved between workers at various levels in the organization, for example from machine-generated, web-created, or front-line customer service personnel (state := *new*) to first or second tier customer service or technical support personnel (state := *assigned* then *opened* then *resolved*) then on to a team leader or even back to the customer (state := *closed*).

The last 5-10 years have seen the introduction of vendor software that captures solutions, turning the trouble-shooting task at support centres into one of searching and matching a database of known solutions. At the time of this research, the HTG support organisation was using SolutionDB[28] as its solution storage / knowledge management system. An assessment of SolutionDB's fitness for purpose is provided in the responses to survey questions in Appendix A (commencing on page 358, with specific comments commencing on page 360).

When SolutionDB was introduced, it was reported to have significantly reduced the problem resolution times through application of Consortium for Service Innovation (CSI)[29] Knowledge

confidentiality of HTG in the written work, and some major last minute funding issues (fortunately resolved). The welcome arrival of our third beautiful baby Jasmin was a fabulous blessing in the middle of all that!

[27] CaseDB is an alias for HTG's case tracking software. In October 2001 the CaseDB business was acquired. The acquiring company does not promote CaseDB as a continuing product.

[28] SolutionDB is an alias for HTG's solution tracking i.e. knowledge repository software. In November 2004 the SolutionDB business was acquired. The acquiring company does not promote SolutionDB as a continuing product. SolutionDB is used by approximately 200 staff in HTG's Asia Pacific Support Group alone (data provided with thanks to Julie Gibson).

[29] The CSI is a non-profit alliance of customer service organizations that are working together to solve industry-wide support centre challenges. HTG is a member of the Consortium for Service Innovation (CSI) and is committed to the Knowledge-Centred Support (KCS) initiative.

Management principles. The Knowledge Centreed Support (KCS)[30] initiative at HTG was working to make even more effective and efficient use of the intellectual capital represented by SolutionDB.

Significant organisational inertia was locked up in CaseDB and SolutionDB at HTG, particularly in regard to culture, training, metrics and management reporting[31]. If someone were able to find or design a better case tracking and problem solving information system, integration to the workflow in HTG's support organisation and the management of such a sizeable change would be a major hurdle.

2.4.2 Problems with Incumbent Systems

While attending a SolutionDB training course for KCS coaches, the participants' comments about SolutionDB and CaseDB were recorded, together with the sharing of trouble-shooting knowledge at HTG. Participants identified a number of problems with the incumbent systems:

- Both CaseDB and SolutionDB were extremely slow in HTG Sydney and consequently very frustrating for users;
- The case tracking CaseDB tool was not searchable at all so that past cases that might be similar to the current case could not be easily found or referred to; and
- The search engine for the SolutionDB knowledge base only allowed a subset of the fields to be searched; it returned too many irrelevant false positive hits; there was no boolean or free text search facility; the globally used ontology was dependent on the local culture; and there was no forum for resolving differences in the meaning of terms, hence searching

[30] KCS is a vision sponsored by the CSI (2004) for creating knowledge that empowers the business. The idea is to capture the solutions generated by the customer support process, and make them available for reuse throughout the support organization. In theory, this allows the support organisation to harness and leverage the knowledge of its employees and customers to improve both quality and efficiency, and scale the business, while effectively managing costs and resources.

[31] As an example of this organisational inertia, both CaseDB and SolutionDB have been acquired by other businesses in the last 5 years. Although the acquirers no longer promote CaseDB or SolutionDB to their new clients and are scaling back their support for these products, at the time of this research HTG had not transitioned to new product offerings either from the acquiring businesses or from their competitors.

with a particular setting in one field might ambiguously exclude solutions that should have been included.

Because the system was so slow, users were reluctant to search SolutionDB for solutions, they were reluctant to create new solutions, and they were reluctant to link their CaseDB cases to existing SolutionDB solutions. Fortunately the performance issues have been addressed to some degree in the last 2.5 years and these last three problems have to some extent subsided.

As well, users complained that there were many duplicate solutions as well as junk solutions. A working group was set up to address this problem by *scrubbing* the knowledge base and vetting all new solutions. This activity provided a significant improvement to the credibility and effectiveness of SolutionDB as a knowledge storage solution.

Finally, it was highlighted that throughout the organisation, many different and incompatible case tracking and knowledge management solutions were used. Whereas the support centre used SolutionDB and CaseDB, engineering used SolverDB[32] and REMEDY[33], and others within the organisation used DDTS[34]. Restricted access and know-how meant that there was very limited cross-divisional learning between different functional groups within the same organisation for the same classes of problem.

2.4.3 The Knowledge Gap

There was a large gap into which knowledge was falling in the HTG support centre workflow. As discussed previously, incoming cases were tracked in the CaseDB, and solutions were (sometimes) stored in SolutionDB. The separation between these two types of vendor offerings was creating significant workflow inefficiency for the support centre.

Firstly, many of the problem attributes captured at the problem tracking stage, were exactly the attributes required to recognise that the user had a particular class of problem on hand, and consequently that a particular type of solution will apply. The double handling of these

[32] SolverDB is an alias for the solution tracking tool used by HTG's Engineering group.

[33] Remedy is another solution tracking tool – see http://www.remedy.com/solutions/servicemgmt/help_desk.htm

[34] DDTS can be used as both a case and solution tracking tool – see http://www-306.ibm.com/software/awdtools/clearddts/version4-7.html

attributes on entry to CaseDB, and again in SolutionDB, left significant room for error, and resulted in significant frustration for support centre personnel.

Secondly, the time consuming process of re-discovering the right set of questions to "work up the case" and hence identify the problem on hand, and the time consuming process of re-discovering where and how to search for a solution, appeared to be repeated many times across the global support organisation. Trouble-shooters had great difficulty in identifying the problem on hand, and finding a relevant solution.

The problem solving expertise i.e. the reasons an expert classifies a case a certain way, were seldom being articulated and shared at the HTG support centre. This same problem was observed a decade ago in the pathology domain – "experts manually interpreting reports are not usually called upon to identify the exact features of the data that led them to their conclusions" (Edwards, 1996, p114).

2.4.4 Time to Fix Repeat Incidents

A brief analysis was undertaken by Jim Mullins at HTG to determine whether the same types of problems were seen multiple times, and if so, the time taken to solve those problems on each repeat occurrence. As shown in Figure 1 (courtesy of Jim Mullins), a number of solutions (labelled A-G) were captured, documented and stored in the SolutionDB knowledge-base. The figure shows a count of the number of problem cases where these solutions were used, and how many hours it took for each case to be resolved (y-axis).

For instance, Problem C, took about 30 hours to solve, but once it was solved in the first instance, it took virtually no time to solve again. In contrast, Problem B took around 24 hours to solve the first time, no time the second time, but the five (5) subsequent incidents took at least as long as the first incident. Although both staff and client availability influence this data, on the whole the data indicates that even though the previous solution has been stored in SolutionDB, the knowledge was lost and needed to be rediscovered repeatedly.

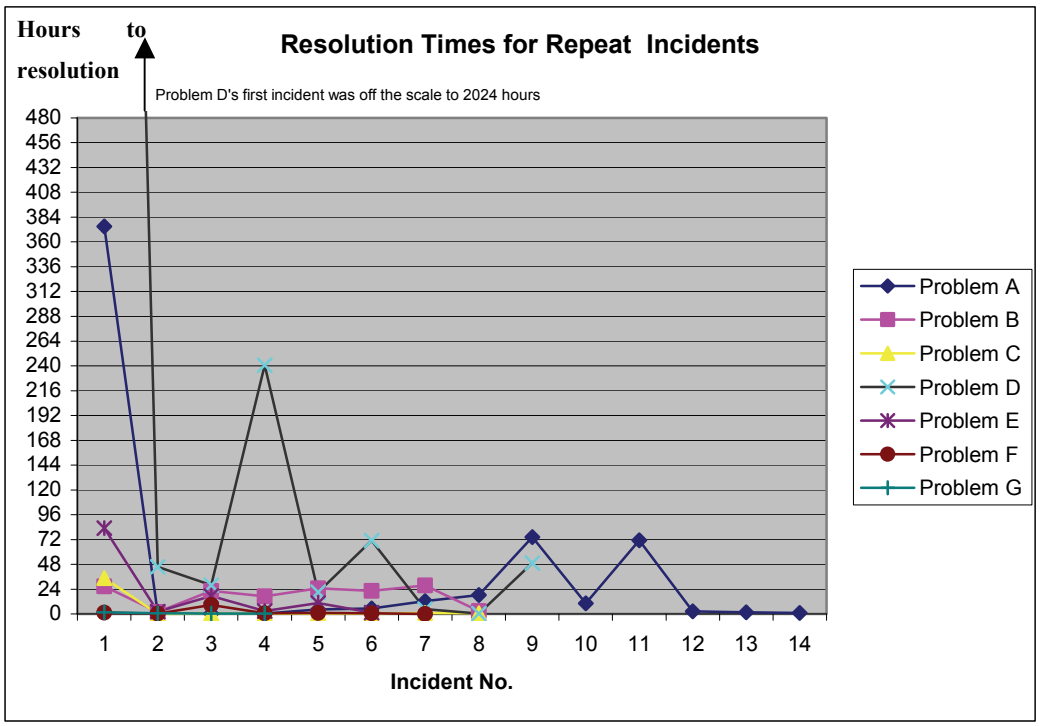
In the first few weeks of the research at HTG, the most common feedback regarding SolutionDB was as follows:

"We can't find old solutions, even the ones we created ourselves!"

(This is further supported by the response to survey question 61 in Appendix A, on page 369.)

Figure 1: Time taken in hours to solve the same problem seen in multiple incidents.

Data Source: "Solution Reuse Value Analysis" from KCS Coach Training



2.5 Support Centre Survey

2.5.1 Survey Design

To examine further what the health check interviews and observations indicated, a comprehensive survey of the HTG Sydney Support Centre was conducted. The survey had the following goals:

1. To determine the current tools and techniques for resolving customer problems;
2. To examine the pros and cons of the current work practice; and
3. To examine future opportunities for improving the trouble-shooting process.

The survey contained two parts: Part A included 67 questions and took participants on average 50 minutes to complete and Part B included 11 questions and took participants on average 5 minutes to complete. A summary of the results to the 78 questions is provided in the text that follows. Please refer to Appendix A (commencing on p 305) for the complete results of this survey.

The products supported by the call centre were divided into two different product groupings: Product Group A and Product Group B. As well, there were two levels of support: Level One Support which attempts to solve the problem quickly at the time of the call or within the first few hours; and Level Two Support who takes over problems that could not be solved by Level One within a reasonable time.

There were 18 respondents in Survey Part A, and 20 respondents in Survey Part B as described in Appendix A Questions 1-7 (page 306). The study covered support staff from both support levels and both product groupings, resulting in 4 groups as shown in Table 2.

Table 2: Number of surveyed Support Staff in Survey Part A

No. Survey Participants	Product Group A	Product Group B
Level One	4	5
Level Two	7	2

Table 3 and Table 4 show the average industry and company experience of the surveyed support staff.

Table 3: Average Industry Experience of surveyed Support Staff

	Industry Experience	Self-assessed Industry Experience relevant to HTG role.
Level One	7 years	4 years
Level Two	12 years	11 years

Table 4: Average Company Experience of surveyed Support Staff

Average Company Experience	Product Group A	Product Group B
Level One	10 months	2.5 years
Level Two	4 years	4 years

2.5.2 Summary of Survey Results

In support of the idea that knowledge re-use is worth pursing in the support centre, on average, survey respondents thought that:

- 64% of customer problems assigned to them had been previously seen by themselves or others i.e. they were repeat problems (see question 14, p 318).
- 76% of customer problems assigned to them would be seen again by themselves or others within the organisation (see question 15, p 319).

From this we can deduce that:

- around 12% of problems were "first-time" problems that would reoccur, so there is a steady flow of new types of problems coming in.
- 24% of problems had never been seen before, and would probably never be seen again, but even for these problems, guided and context relevant top-level trouble-shooting information could either support a quicker time to resolution, or at a minimum indicate that a solution was not yet known[35].

Additionally:

- 81% of solutions applied by Level 1 respondents are replicated solutions whereas 44% of solutions created and applied by Level 2 respondents are replicated solutions i.e. someone has conceived of this solution before (see question 16, p 321).
- When a customer problem was assigned to them, in 67% of cases trouble-shooters would **not** know the solution straight away and would need to refer to other sources of information to solve the case (see question 17, p 322).
- Respondents would involve their team-mates or others in 43% of the problem cases assigned to them (see question 12, p 315). As well, communications (see question 24 on page 331 and question 28 on page 334), meetings (see question 25 on page 332) and collaborative processes (see question 26 on page 332) were identified as important components of the trouble-shooting process.

[35] The importance of being able to fall back on a layer of higher-level general knowledge when more detailed and specific knowledge is missing, is also expressed by Dazeley and Kang (2004, p2).

- When asked at what point in time trouble-shooters would try to get help from their team-mates or others, respondents describing their level of expertise as *novice* through to *senior* would wait up to 2 hours, whereas those describing their expertise as *expert* would wait from 2-4 hours to up to 2 days depending on the problem area and impact / severity (see question 13, page 318).

The diversity of resources used to solve problems at HTG are described in questions 19-23 (commencing on page 324). Table 5 provides an ordering of some of these problem solving resources.

Table 5: Most to least frequently relied upon resources when solving problems

The mechanism used to generate the linear weightings for the Likert Scale applied here is discussed in Appendix A, Question 20 on page 325.

Resource	Weighted Score
My problem solving skills	84
My knowledge	83
Knowledge Base Tool	75
Discussion with peers	71
My experience in the field	62
Internet searching	59
Engineering presentation documents	57
Engineering Technical Manuals	57
Training course handouts	54
External Corporate Knowledge Bases eg Microsoft, Sun	51
Customer Manuals	50
Case Tracking Tool	43
Emails	27
Level 2 Troubleshooting Documentation	21

Note that these results are consistent with those reported by Shaw and Gaines (2000, p1): "*developing knowledge-based system involves knowledge acquisition from a diversity of sources, often geographically distributed. The sources include books, papers, manuals, videos of expert performance, transcripts of protocols and interviews, and human and computer interaction with experts.*"

As well, for at least 48% of the mundane routine repetitive types of problems that had been seen before, and that would be seen again, respondents indicated that they would need to refer

to other sources of information to solve the problem. These mundane problems comprised 64% of the case load!

In answer to "What are the biggest roadblocks that stop you being effective and efficient in your role?" the following responses were received (see question 29, p 335):

- Training / Knowledge (8 responses);
- Accessible documentation (6 responses);
- Solution database (3 responses);
- Time pressures (3 responses);
- Escalations - these are where a problem is referred up the trouble-shooting chain for example from level 1 support to level 2 support (2 responses);
- Customer contact (2 responses).

Despite a comprehensive training program at HTG, training was seen by trouble-shooters to be inadequate (see survey question 29 on page 335). When asked "Have you had all the training that you require to perform your trouble-shooting role effectively and efficiently?" 18% said yes, 23% were undecided and 59% said no (see question 31a, p 338).

This prompted me to ask why training was inadequate despite the organisation's extensive training program. The answer appeared to be that the solution space was very complex, involving a range of different knowledge types and sources. What was important to trouble-shooters was their ability to access the smallest complete set of relevant information that would help them solve the problem on hand. Access to relevant and current technical information was particularly problematic (see questions 32a and 32b commencing on page 341).

Survey results confirmed that there was a large gap into which knowledge was falling in the HTG support centre workflow. The separation between CaseDB and SolutionDB was creating significant workflow inefficiency for the support centre (see question 36a on page 348, question 36b on page 349, and question 37 on page 350).

2.5.3 The Nature of Trouble-shooting

When asked how they actually solve problems, trouble-shooters comprehensively detailed their personal processes of gathering information about the case on hand, using that

information to classifying the problem, and matching the classified problem to relevant solutions. This matches the *case-configuration-classification-conclusion* cycle proposed by Clancey (1984) and further discussed in (Clancey, 1985, pp 27-29 and pp 49-51). The responses have been summarised as follows (see also Question 11 on page 313):

1. Find out what the problem is (i.e. classify the problem). Listen and understand the customer's problem, their concerns, and the impact of the problem. Gather as much relevant information as possible. Asking questions to try to narrow the problem (i.e. work-up and configure the problem case).
2. Review the case including the past efforts of the customer and HTG. Try to recreate / replicate the problem.
3. Check for known bugs and / or symptoms and / or related issues.
4. Attack the problem on multiple fronts if possible i.e. be prepared to try several different approaches and engage several different resources for information. Repeat from step 1 as needed.

The responses from the 18 trouble-shooters from two different support levels and product groupings at HTG indicated that trouble-shooting includes the ability to configure (i.e. work-up) a case, classify it, and locate its relevant solution. As well, multiple classifications may be relevant to a problem, and several different solutions may need to be tried[36].

The observation that trouble-shooting includes the ability to configure a case and then classify it is not new. Clancey offers (1984): "*the essence of classification problem solving is that the solution to a problem is selected from a list of pre-enumerated possible solutions*" and that construction "*involves piecing together a solution*".

2.6 Requirements Analysis

In light of the need to satisfy their own, their customers, and their shareholders needs, the problem for trouble-shooters at the support centre appeared to be about finding the right solution in the right context. Trouble-shooters were under significant time pressure (see

[36] Systems can fail for multiple different reasons, and problems can co-occur. Hence the troubleshooting domain at HTG was a multiple classification problem domain. An excellent overview of the multitude of ways in which systems can fail is provided in Perry (1995).

question 38, page 351) to solve problems; they relied on many diverse sources of information as well as the trouble-shooting know-how of their peers. They were constantly challenged by an overload of information and a lingering doubt about the appropriateness of a given solution to the problem on hand (described in Researcher's Note 14 on page 332 and supported by responses to survey question 51 on page 361).

The health check interviews, observations, and survey culminated in the development of a set of requirements for an improved trouble-shooting software platform. Although it was beyond the scope of this research to address all of these requirements in the proposed prototype system and software trial, this requirements list is included in the thesis in Appendix C (page 390) to demonstrate the broader context in which the knowledge representation and knowledge acquisition algorithms (proposed later on) must work.

2.7 Chapter Summary

In this chapter, the findings of the interviews, observations and survey at HTG were presented. As well, the requirements for an improved trouble-shooting information system were collated and presented in Appendix C.

In answer to the research question:

Q1. What does the HTG support centre environment tell us about the nature of trouble-shooting?

the interviews and survey revealed that trouble-shooting in the ICT domain at the HTG support centre involves complex problems, requiring a wide span of experience and a wide span of knowledge from multiple different and global sources. At HTG there were limited and disparate forums in which trouble-shooters could share that knowledge, for instance in training sessions, procedures manuals, on a wide variety of separate and unsearchable Intranet sites, or in face-to-face meetings.

Users found it difficult to find previously created solutions in the organisation's knowledge base. The domain was jargon rich and many solutions didn't lend themselves to free text keyword searches. Boolean search was not available in the solution database, and the case-tracking database had no search facility at all. Separate parts of the organisation had completely separate databases for tracking problems and their solutions.

Individual tacit knowledge that was seldom being made explicit and shared with the trouble-shooting group included: the problem determination, where-to-search and what-to-search-for knowledge required for specific problem classes. There was no workflow mechanism for more experienced users to rapidly share problem configuration or classification knowledge with their colleagues. As well, there were limited paper-based guided trouble-shooting mechanisms to help novices come up with the right questions that would enable them to distinguish the particular class of problem on hand and hence find its solution.

Although it is difficult to quantify, it seems that at HTG, organisational learning of new problem classes was slowed by the existing information dissemination systems and the organisation was relatively inelastic[37] and unresponsive to new problem classes.

In summary, as a result of the interviews, observations and survey it was found that trouble-shooting includes the ability to configure (i.e. work-up) a case, classify it, and locate its relevant solution. Problem solving appeared to follow a case-configuration-classification-conclusion cycle. As well, multiple classifications may apply for a problem, and several different solutions may need to be tried. It was found that problem solving required diverse types and sources of information, and that much of the case-classification matching knowledge was not shared in the support centre, but rather stored in the minds of individual experts.

The next chapter reviews some other KM approaches that have influenced the design of the trouble-shooting solution proposed by this research.

[37] meaning slow or unable to adapt and learn

CHAPTER 3: OTHER KM APPROACHES

3.1 Chapter Outline

As previously presented in section 1.3 on page 3, the aim of this research was to:

> *Develop a knowledge acquisition approach that will facilitate the capture and recall of trouble-shooting know-how, in an environment where problems and their solutions are continuously evolving, and where a distributed group of stakeholders can both contribute to and benefit from the acquired knowledge*.

Fields of research related to this aim include information retrieval (IR), decision support systems (DSS), knowledge discovery in databases (KDD), artificial intelligence (AI), machine learning (ML) and data mining (DM). For the sake of brevity, the description and discussion of these research fields has been relegated to Appendix D (commencing on page 398).

Many vendors supplying support centre workflow software boast the use of AI, ML and DM techniques in their Knowledge Management (KM) product offerings. Indeed, when challenged with the task of improving the effectiveness and efficiency of trouble-shooting at HTG, the use of data mining was initially suggested. In this chapter, some of the reasons why traditional DM and hence AI and ML techniques have limited ability to solve the above problem are presented.

Similarly, support centre staff initially suggested the use of a Google-style search engine to search HTG's internal corporate databases. So in this chapter, some of the problems with applying popular search engine technology to the trouble-shooting task are highlighted.

In years gone by, traditional expert systems, including rule-based and case-based reasoning systems have been used for information recall in numerous business and scientific domains. More recently, ontologies have been adopted as a way to share, reuse and process domain knowledge. In summary then, this chapter addresses the following research question:

> *Q2. What are some of the problems with existing knowledge management approaches in domains such as the ICT support centre?*

This chapter adds to the analysis provided in section 2.4 on page 17 of HTG's incumbent case and solution tracking tools.

3.2 Data Mining

The suggested Data Mining (DM) efforts to develop a predictive classification model that would link incoming problems to their relevant solutions at HTG by discovering associations in the existing databases, and hence provide a platform for searching, would be thwarted by numerous problems. The majority of DM[38] techniques require that the exemplar cases be described as sets of attribute-values pairs (Dunham, 2003)[39]. As well, DM techniques require that cost functions be defined that can operate on the attribute-value pairs to determine similarity or dissimilarity between the incoming cases and the outgoing problem classifications. What was found at HTG was that neither the case-tracking database nor the solution-tracking database contained enough domain knowledge to easily identify the important attributes of the case, or allow suitable similarity / dissimilarity measures to be developed.

There were very limited records of which solutions had previously been applied to cases, so that supervised learning techniques were impossible. As for unsupervised learning techniques, for the present set of cases it was impossible to come up with a universal similarity or dissimilarity measure that would enable clusters of cases and their corresponding classifications to be discovered. As demonstrated in Appendix B (commencing on page 387), the case data was numerically rich and functionally complex.

Efforts to data mine the existing databases would have been unnecessarily confused by the errors learnt from the historical training data. The data that existed fundamentally lacked any indication of the attributes or rules that meant that a particular solution should apply to a particular class of problem – this data was falling through the gap between the CaseDB case tracking and SolutionDB solution tracking databases. The knowledge of how to get from the case on hand, to its solution simply didn't exist in the databases, and couldn't be rediscovered. Rather, it was tacit knowledge, accumulated and stored in trouble-shooter's heads, within the organisation. Some interpretations of the term "tacit knowledge" relevant to this research are

[38] For those unfamiliar with Data Mining, a brief description has been provided in Appendix D.6, commencing on page 400.

[39] Recent advances have begun to allow other representations including trees, graphs and relational representations such as first order predicates.

provided in Appendix E (commencing on p 404). More general problems associated with data mining are listed in section D.8 on page 403.

3.3 Popular Search Technologies

According to Alexa[40], the top three Internet sites globally in terms of popularity are Yahoo[41], MSN[42] and Google[43] (Alexa, May 2006). Each site makes a perfect home page by offering an Internet search engine plus news portal, and hence provides a launch pad to the modern information-at-your-fingertips world. Several of the staff at the HTG support centre suggested that the company install an enterprise search engine, like that marketed by Google, inside the support centre's firewall to assist with solution search. However, conventional web search engines suffer from several problems (adapted from Dunhan, 2003, p41):

- Abundance – too much information[44];
- Limited coverage – caching and elimination is required to handle the size of the search space leading to outdated and possibly incomplete search results;
- Limited query – most search engines provide access based on only simple keyword based searching; and
- Limited customisation – the desired results are often dependent on the background and motivation of the searcher – at this stage customisation based on user profiles or historical information is only provided by some of the more advanced search engines.

As described later on (section 5.3 on page 71) the most valuable information to popular search engines is the *anchor text*[45] applied by other independent webmasters on the Internet to refer

[40] http://www.alexa.com/site/ds/top_sites?ts_mode=global&lang=none

[41] http://www.yahoo.com/

[42] http://www.msn.com/

[43] http://www.google.com.au/

[44] Many users experience information overload when searching for answers to their questions. The problem isn't a lack of information, but rather how to find, sort, filter and assimilate the information you need (CSIRO website, Jan 2005).

[45] For a definition of Anchor Text, please see the glossary (page 277).

to a given website. Further to this, some of the reasons why current popular search approaches (such as Google, Yahoo and MSN) may not work well on their own when searching for solutions in the ICT domain are as follows:

- Anchor-text simply does not exist for enterprise case-tracking and solution-tracking databases.
- The ICT domain is notorious for being jargon-rich and simple keyword searching may not be sufficient to locate the relevant information. A richer grep/sed style pattern matching mechanism may be needed, for example to match sequences of hexadecimal error codes, or patterns of version number, and at times using strategically placed wildcards. The example case in Appendix B (commencing on page 387) exemplifies this problem.
- In addition to boolean operators, the search may need to handle numerical operators such as <, > etc.
- The data may already be somewhat structured, and the information abundance problem would be greatly alleviated by constraining search queries to within fields provided by the existing data structures. For example a pathology case may contain structured data relating to the concentration of certain chemicals, pathogens or hormones in the sample.
- Custom ontologies may be required to define organisation and domain specific jargon, for example that 'version' is the same as 'release' or 'edition' or 'issue' but not the same as 'iteration'.
- Current information is needed as real-time data, as opposed to cached, and therefore non-current data.

In summary, popular search engines rely on collaboratively constructed anchor-text to index their vast search domains. Currently their usefulness is limited to plain text searching. When it comes to more complex (e.g. numeric) data, a different and richer query mechanism is required. In the trouble-shooting domain, closing the loop between the answers a user is searching for, and the quality of the answers retrieved is currently a vital but missing step.

3.4 Ontologies

In recent years, ontologies have been adopted in many business and scientific communities as a way to share, reuse and process domain knowledge[46]. An ontology describes the concepts and relationships that are important in a particular domain, providing a vocabulary for that domain as well as the meaning of terms used in the vocabulary[47]. Ontologies range from taxonomies and classifications, database schemas, to fully axiomatized theories. Some practical examples of ontology usage are provided in Farquhar et al. (1995b, p29). Tran et. al (2006) provide a further insight to the development of ontologies in their work on agent oriented software engineering (AOSE), and emphasise that "*an ontology should capture consensual knowledge that is not restricted to an individual, but is shared and accepted by a group*" (Tran et. al, 2006, p3).

Data mining activities in the semantic web have enabled ontologies to be automatically discovered in text, however the problem of making the ontologies human readable and usable, and hence the problem of acquiring useful ontological knowledge from human experts remains.

Perhaps an even greater problem in ontology usage is how to identify and resolve semantic conflicts in disparate knowledge bases, built for similar but different audiences in similar but different domains. A subset of this problem is how to identify and resolve semantic conflicts in a shared knowledge base, built for individuals that are collaborating in a shared domain. An extension of these collaboration issues is provided in Appendix F (commencing on p 405). In section 13.7 on page 261 I describe how the system proposed by this research might be used to address this problem in the future.

Ontologies are great at specifying class attributes, behaviours and relationships, however for case-based problem solving a specific focus is required on how to recognise when a particular case belongs to a given class in the ontology. The features used to recognise when an instance belongs to a given class are not necessarily the features that you would use to provide

[46] see http://protege.stanford.edu/overview/

[47] See Gruber (1993) and also http://en.wikipedia.org/wiki/Ontology_%28computer_science%29.

an ontological definition of that class[48]. The fact that species identification is different to species classification is well known for instance in the natural sciences[49]. Classifications will tend to describe the most common properties and behaviours of the class, whereas identifications need to handle a range of (polymorphic[50] and hence exception related) deviations from normal.

In attempting to match a known case-on-hand with a class in the ontological database, users have to make the same mental leaps as trouble-shooters about the nature of the problem on hand, where-to-search and what-to-search-for. As well, users need to know what extra information is relevant for them to work-up a case and locate more specific knowledge. Identifying the minimum complete set of rules that will allow a user to select the appropriate ontological entry (i.e. class) requires a knowledge acquisition activity all of its own.

3.5 Knowledge Acquisition and Expert Systems

Knowledge Acquisition (KA) is the process of acquiring knowledge from one or more third parties, be it from some individual, from a machine, or from a group of individuals or machines. Clancey (1984, p1) describes several steps in KA including problem selection, codification, testing and refinement. Shaw and Gaines provide the following definition (1989):

> "*Knowledge acquisition is essentially a negotiation process leading to approximations to conceptual structures… adequate for some practical purpose…*"

[48] For example, someone who "swims like a brick" is an instance of a "poor swimmer", but it is doubtful that the subclass "poor swimmer" would have a "swims like a brick" attribute or behaviour in its ontological definition.

[49] For instance, in the biological, botanical, ecological and zoological domains.

[50] Polymorphism is frequently referred to in object oriented knowledge representation systems. It refers to the situation where a child class inherits many of the general properties of a parent class, with some notable and specific exceptions. For example: birds generally fly; penguins are birds; penguins don't fly. This example is derived from Yao et. al (2005) and is referred to again in section 4.3.2.4 (page 60).

Throughout the 1980s the Expert System[51] became the repository of accumulated knowledge regarding possible error scenarios and how they could be resolved[52]. According to Edwards, experts are good at filtering out irrelevant information in order to get at the basic issues. Expert systems reflect this practice by reasoning with shallow (empirical or heuristic) knowledge (Edwards, 1996, p37).

3.5.1 Conventional Expert Systems

Conventional expert systems are typically rule-based reasoning (RBR) systems. Users can add RuleNodes directly to the rule tree and (subject to the implementation) even edit, move and delete them. Edwards suggests that in conventional KA systems, the expert attempts to reconcile many contexts for a given classification by asking: "*what are all the contexts for this classification*". He observes that "de-contextualising knowledge" comes at the expense of the local context and hence the usefulness of the accumulated information. Edwards offers the insight that: "*conventional expert systems give primacy to the classification, rather than the context"* (Edwards, 1996, pp 170 - 171). Hence conventional expert systems suffered from such problems as giving poor or wrong advice when queried outside their very narrow domain (the brittleness problem[53]) and the lack of a clearly defined relationship between the knowledge in the knowledge base and the problem cases reported (Richards, 1998a).

As well, conventional single expert systems were not designed to handle the continual evolution of knowledge. Thus, KA and its subsequent maintenance became a bottleneck in the management and reuse of knowledge (Feigenbaum, 1980). As described previously (section 2.3.3 on page 16; section 2.5.2 on page 24; and Appendix A[54]), the HTG interview and survey

[51] The 20 questions website at http://www.20q.com and "guess the dictator or sit-com character" website at http://www.smalltime.com/dictator.html are interesting self-learning expert system programs that demonstrate the gathering of information from the Internet community using "toy" KA problems (Chklovski, 2001).

[52] This text appears in (Richards and Vazey, 2004)

[53] This is also termed the "closed world" assumption in the RULE-ML / Semantic Web community.

[54] The HTG support centre survey reported in Appendix A showed that trouble-shooters would involve their team-mates or others in 43% of the problem cases assigned to them (question 12, p 315). As well, communications (question 24 on page 331 and question 28 on page 334), meetings (question 25 on page 332) and collaborative processes (question 26 on page 332) were identified as important components of the trouble-shooting process.

results showed that trouble-shooting knowledge is learned, acquired, generated and consumed in a decentralized manner by multiple stakeholders and using multiple sources of information[55] across the global support organization. Centralising the control of such knowledge through conventional rule based systems or formal ontologies may create greater consistency of expression, at the expense of currency, relevance, and accuracy.

3.5.2 Case-Based Reasoning (CBR) systems

Case-Based Reasoning (CBR) aims to help users solve problems by adapting previously successful solutions to similar problems (Marir and Watson, 1994). It is a cyclical process comprising the four 'R's of *retrieving* the most similar case, *reusing* the case to attempt to solve the problem on hand, *revising* the proposed solution if necessary, and *retaining* the new solution as a part of a new case (Aamodt and Plaza, 1994). CBR is appropriate where there is no formalized knowledge in the domain or where it is difficult for the expert to express their expertise in the format of rules (Kang et. al, 1996).

CBR requires that useful vectors of attribute-value (A-V) pairs appropriately partition the domain. Hence one of the problems for CBR in new and dynamic domains is that the wise selection of A-V pairs to partition the domain is not at all easy, and it is often an ongoing process (Kang, 1995, p53). Further to this, Pearce at. al. (1992) reported on the use of case-based reasoning for managing a large case-based library of architectural designs. The *Archie* system used nearest-neighbour matching and model-based clustering to retrieve cases (p17). Hence part of the background knowledge required by *Archie* was the functions that it could apply to the attribute vector of each case to determine its similarity or dissimilarity to other cases in the system. What was learnt was that real-world cases are often incomplete, and that this severely limits the usefulness of past cases in making new design decisions (p19). As well, real-world cases are very large – the domain model requires a huge number of attributes to be defined, into which the case can and should be decomposed (p19). Finally, users want to see the knowledge in multiple different views, and they want to be able to easily navigate between the various views.

[55] Many different knowledge resources were used to solve problems (refer to survey questions 19-23, commencing on page 324). Table 5 (page 25) provided a summary of some of these solution resources.

Mansar and Marir (2003) suggested the use of a case-based reasoning (CBR) technique for Business Process Redesign (BPR). They highlighted the limitations of the CBR approach: the manner in which cases are indexed is not well-defined. In their BPR example, users were manually required to order cases into a case hierarchy that presented as somewhat of a maintenance nightmare. In their proposed system, the manner in which cases were to be indexed, and the manner in which similar cases should be identified was undefined.

As highlighted by Kang (1995, p51), the goal of CBR is not to find knowledge in the knowledge base that applies to the present problem, but to find a case similar to the current case in a database of cases. This is significant, since a user may find a similar case, and yet find that they are no closer to recognising an appropriate solution.

As mentioned previously in section 1.3 on page 3, a sub-goal of this research was to explore the possibility to adapt Ripple Down Rules (RDR)[56] to the support centre context. Ripple Down Rules can be considered as an augmented CBR KA approach that combines the strengths of CBR and RBR. It reduces the indexing and maintenance problem of former CBR techniques by providing a mechanism for incrementally acquiring suitable rule-based indexes for common classes of cases. The technique asks experts that are intimately knowledgeable of the case context to enter heuristic rules that are used to determine when a conclusion applies to a given case. Amongst others, Beydoun et. al (2007) recognise RDR as a framework for ontology acquisition.

3.6 Chapter Summary

In answer to the research question:

Q2. What are some of the problems with existing knowledge management approaches in domains such as the ICT support centre?

and in addition to the analysis provided in section 2.4 on page 17 of HTG's incumbent case and solution tracking tools, this chapter described the numerous problems that exist with data mining techniques, popular search engines, ontology and conventional expert system approaches to knowledge management in the ICT support centre domain.

[56] The family of Ripple Down Rules (RDR) algorithms includes Single Classification (SC) and Multiple Classification (MC) RDR.

At the commencement of the research there was no suitable domain model for the incoming cases, no suitable similarity / dissimilarity measures to determine appropriate classification clusters, and no record of the links between different classes of problems and their solutions. The missing link seemed to be the capture of relevant and otherwise unspoken trouble-shooting knowledge from human experts that could be used to guide trouble-shooters in quickly working-up their cases, classifying them, and locating the relevant solutions.

Data Mining (DM), Case Based Reasoning (CBR), and Ripple Down Rules (RDR) techniques each require that the user create a model of the domain, comprised of a vector of attributes that can be used to identify each case in the system, and a set of functions or rules that can be used to determine the similarity or differences between cases in the system. The advantage that RDR proponents claim over the former two methods is the ease with which RDR is able to extract this domain and classification knowledge from its users. RDR is described in greater detail in the next chapter.

CHAPTER 4: RIPPLE DOWN RULES

4.1 Chapter Outline

In this chapter, the Ripple Down Rules (RDR) algorithm and data structure is explored in detail, and its main implementations: Single Classification Ripple Down Rules (SCRDR) and Multiple Classification Ripple Down Rules (MCRDR) are described along with their strengths and limitations.

4.2 Introduction to RDR

RDR supports the incremental capture of a set of classification rules in the specific context of the case being classified. The philosophy underpinning RDR is that knowledge is socially situated, contextual, continually changing, and emergent (Compton and Jansen 1989). One noteworthy feature of RDR is that it attempts to acquire tacit knowledge (described previously in Appendix E, page 404). In other words it aims to capture otherwise unspoken knowledge as domain experts directly interact with cases. The end result is that heuristic knowledge that might have been difficult to articulate without the context of a specific case becomes articulated and codified, a process that Nonaka, Takeuchi and Umemoto (1996) refer to as externalisation.

The RDR approach was developed in response to the difficulties associated with acquiring and maintaining a rule-based system (Compton and Jansen, 1989). It was observed that pathology experts had difficulty describing what they knew but that they were good at looking at a case and saying how they would handle it[57]. When asked why a certain conclusion applies in a given situation, an expert generally does not explain how the conclusion was reached but, rather, gives a justification for why the conclusion is correct (Compton and Jansen, 1989) (Kang, Gambetta, Compton, 1996, p258). The idea is that experts are good at judging cases, but not good at providing knowledge in abstract forms (Manago and Kodratoff, 1987; in Kang, 1995 p10).

The RDR approach is based on a situated view of cognition which sees knowledge as something made up to fit the situation and always evolving, resulting in the use of cases to

[57] Some of this text appears in (Richards and Vazey, 2004).

provide context and an exception structure to support local patching of the knowledge[58]. This view is in keeping with others in the knowledge management field such as Sternberg (1995) who uses work-place scenarios to measure tacit knowledge and Stenmark and Lindgren (2003) who suggest that knowledge be captured during normal work processes[59]. In contrast to conventional rule-based-reasoning (RBR) expert systems, it could be said that RDR-based systems give primacy to the context, rather than the classification.

RDR offers the pragmatic realisation that experience-based learning involves asking questions on a case-by-case basis about the similarities of and differences between things. The RDR philosophy recognises that humans are amazingly adept, and usually better equipped than computers, at recognising whether two cases belong to the same or different classifications, and whether a set of conclusions fits a given classification (Compton and Jansen 1989).

While the need for knowledge engineering expertise to assist with initial modelling of the domain is acknowledged, RDR proponents argue that the simplicity of the approach allows domain experts to be responsible for entering and maintaining the knowledge (Compton et al 1991) (Richards, 1998a, p16). Strengths of RDR include maintenance and validation performed directly by the domain expert; and the use of cases to provide context and assist knowledge acquisition[60].

4.3 SCRDR and MCRDR

The RDR structure provides a directed acyclic graph that forms a decision tree where each decision node comprises a test of one or more parameters in the incoming case and provides a Boolean output. In the case of SCRDR the exception structure allows a single classification to be derived for the incoming case; and in the case of MCRDR the exception structure allows multiple classifications to be derived for the incoming case.

The following figure presents a sequence of training cases that illustrate the training of some sample RDR knowledge bases. These training cases have been created for an ICT sample domain.

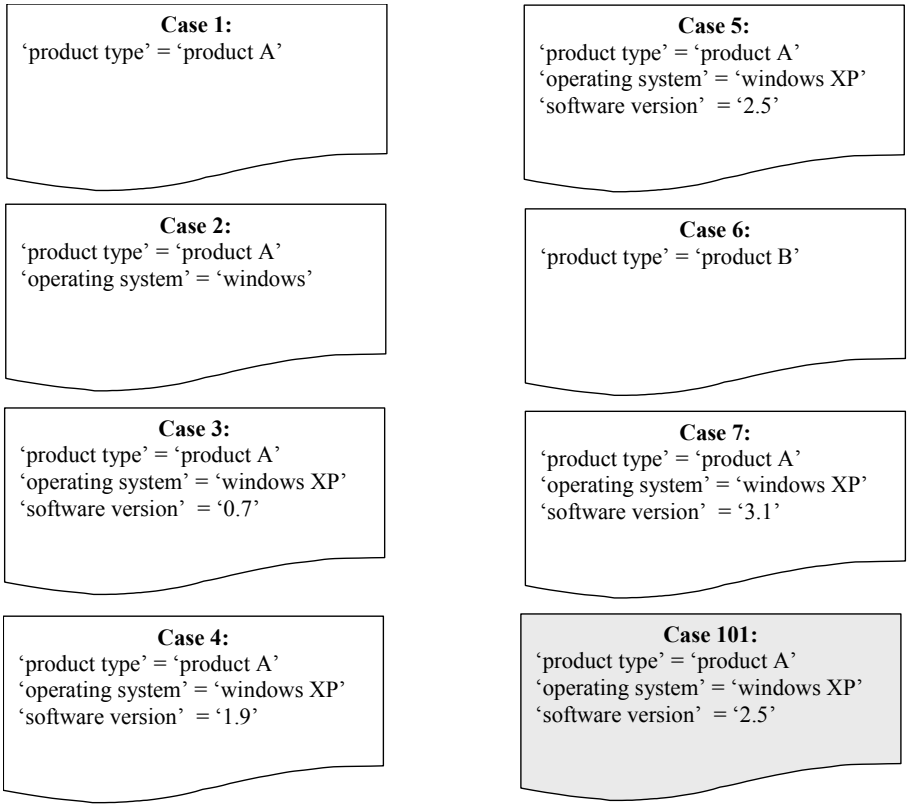
---

[58] Some of this text appears in (Vazey and Richards 2005a).

[59] Some of this text appears in (Vazey and Richards, 2006a).

[60] Some of this text appears in (Vazey and Richards, 2004).

Figure 1: RDR training cases



4.3.1 Single Classification Ripple Down Rules (SCRDR)

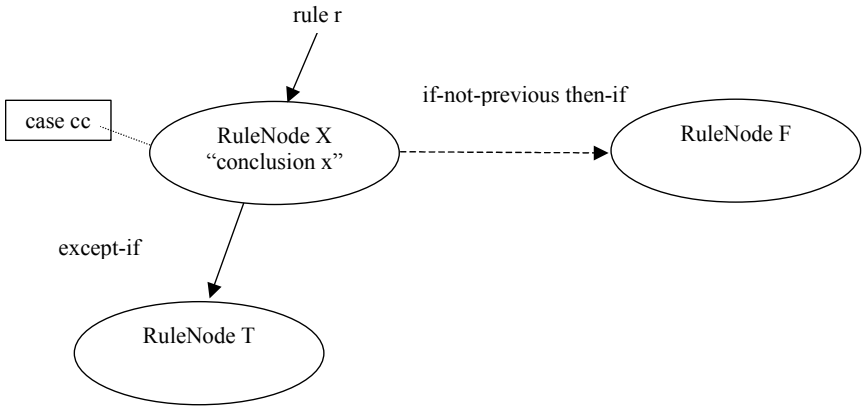
This subsection details structural and algorithmic aspects of the Single Classification Ripple Down Rules (SCRDR) approach. For the sake of expediency, readers familiar with the technical aspects of SCRDR might like to skip forward to section 4.3.1.1 on page 46.

RDR were first developed to handle single classification tasks. An exception structure in the form of a binary tree is used to provide rule pathways (Richards, 1998a, p54). Each RuleNode in the tree is associated with a rule, a conclusion, a cornerstone case[61], possibly one child RuleNode on a true branch, and possibly one child RuleNode on a false branch.

Figure 2 shows the nomenclature used to illustrate this system. Using the grammar developed by Scheffer (1996), an SCRDR can be defined as the set <*r, T, F, cc, x*> where *r* is the rule, *T* is the `TRUE` branch RuleNode, *F* is `FALSE` branch RuleNode, *cc* is the associated cornerstone case, and *x* is the conclusion.

[61] The term *cornerstone case* was coined by Horn et al. in 1984 to refer to cases that have uniquely forced changes to be made to the expert system.

Figure 2: SCRDR RuleNode Nomenclature



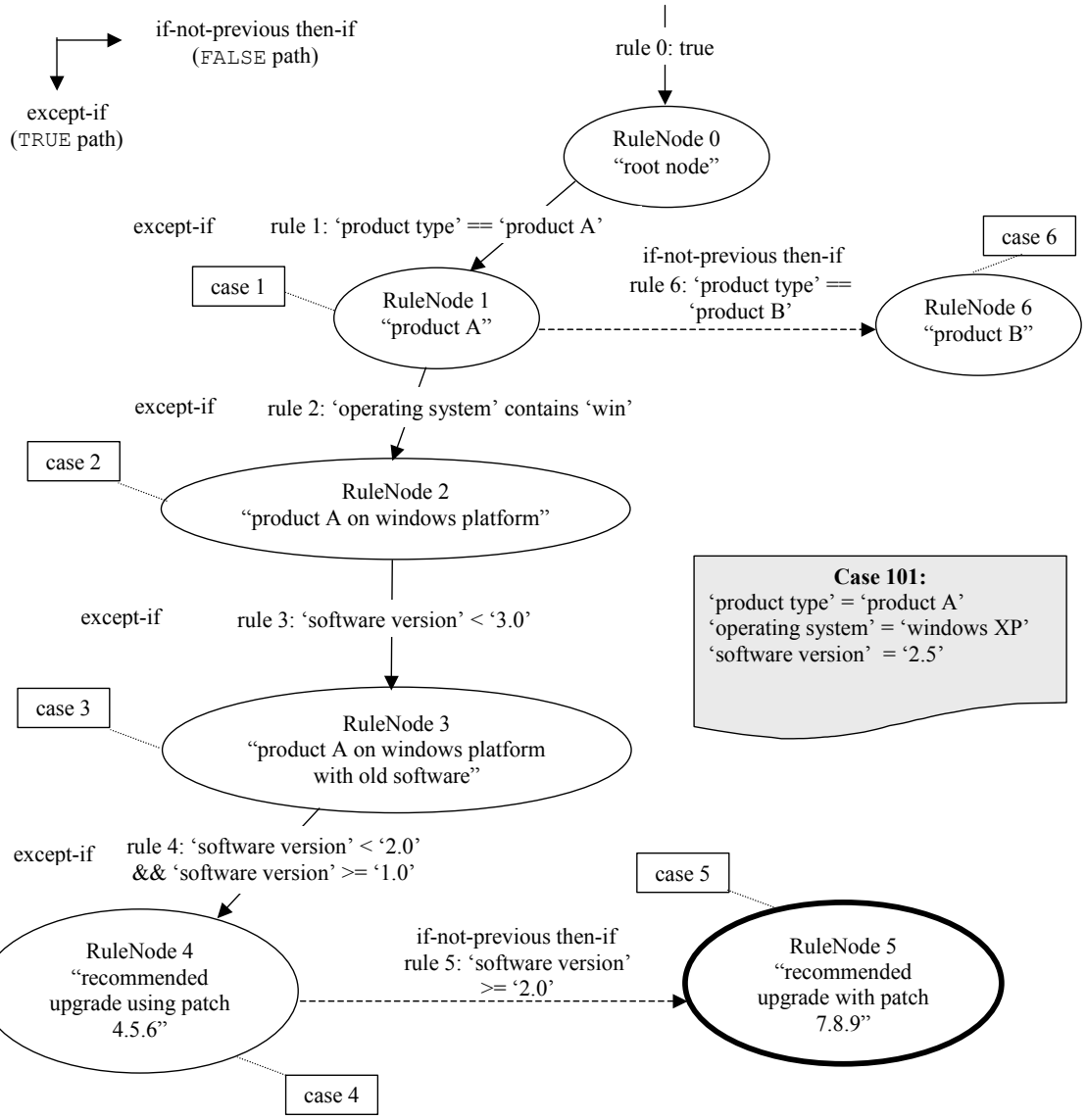
When the expert determines that the conclusion at the last `TRUE` RuleNode is incorrect or inappropriate, in other words that it is an invalid conclusion or a misclassification, a child RuleNode is added. If the rejected RuleNode's rule remains `TRUE` then the child RuleNode is attached to the `TRUE` branch of the rejected RuleNode, otherwise the child RuleNode is attached to the `FALSE` branch of the rejected RuleNode (Richards, 1998a, p54).

The case that caused the misclassification to be identified is stored in association with the new RuleNode and is referred to as the ***cornerstone case***. The purpose of storing the case is to remind the user of the context in which that RuleNode was acquired, and thereby provide validation of the new rule at the new RuleNode (Richards, 1998a, p54). The new rule must distinguish between the new case and the case associated with the RuleNode that gave the wrong conclusion (Compton & Jansen 1990).

In the operation of an SCRDR system, the case is first evaluated against the top-most RuleNode. For each RuleNode, when its rule is satisfied then its `TRUE`-path exception rule is evaluated and its `FALSE`-path rule is ignored. In contrast, for each RuleNode, when its rule is not satisfied its `FALSE`-path is followed and its `TRUE`-path is ignored. The last `TRUE` RuleNode for the resultant path taken through the rule tree gives the final conclusion for the case on hand.

The following figure presents a sample SCRDR knowledge base.

Figure 3: **An SCRDR KBS**



In the above figure, each RuleNode can be seen as a portion of a pathway that leads from itself back to the root RuleNode which is RuleNode 0. The highlighted RuleNode 5 represents the last true RuleNode for Case 101 from the set of training cases in Figure 1 on page 42. Using Case 101 as an example:

- Rule 0 evaluates to `TRUE` for Case 101 so RuleNode 0's `TRUE` (except-if) path is followed. On taking that path,
- Rule 1 evaluates to `TRUE`, RuleNode 1's `TRUE` (except-if) path is followed. On taking that path,

- Rule 2 evaluates to `TRUE` so RuleNode 2's `TRUE` (except-if) path is followed. On taking that path,
- Rule 3 evaluates to `TRUE` so RuleNode 3's `TRUE` (except-if) path is followed. On taking that path,
- Rule 4 evaluates to `FALSE` so RuleNode 4's `FALSE` (if-not-previous then-if) path is followed. On taking that path,
- Rule 5 evaluates to `TRUE`. RuleNode 5 has no child nodes on either its `TRUE` or `FALSE` branches, so for Case 101, the final RDR conclusion is a "recommended upgrade with patch 7.8.9".

The sample SCRDR knowledge base shown in Figure 3 was formed in the following manner:

- At first the only node in the SCRDR tree was the root node, RuleNode 0, with a rule that always returned `TRUE`, and a null, "root node", or default[62] conclusion. All cases presented to the rule tree would initially satisfy this RuleNode.
- Case 1 from the set of training cases in Figure 1 on page 42 was presented to the SCRDR engine. The expert disagreed with the conclusion from RuleNode 0. The expert offered a new rule, shown as rule 1, and RuleNode 1 was created. Case 1 became the cornerstone case for RuleNode 1 since it was the first case that caused RuleNode 1 to be created.
- Case 2 was presented to the SCRDR engine. After evaluating this case against the maturing rule tree, the last `TRUE` RuleNode for the case was RuleNode 1. The expert disagreed with the conclusion from RuleNode 1. Not because the conclusion was `FALSE`, but rather because they wanted to provide a different conclusion, and in this particular case, a more specific conclusion. The expert offered a new rule, shown as rule 2, and RuleNode 2 was created. Case 2 became the cornerstone case for RuleNode 2.

[62] Note that every RuleNode, including the root RuleNode holds the default classification and conclusion for cases that satisfy that RuleNode and none of its child RuleNodes.

- Case 3 was presented to the SCRDR engine. After evaluating this case against the maturing rule tree, the last `TRUE` RuleNode for the case was RuleNode 2. The expert was dissatisfied with the conclusion from RuleNode 2. The expert offered a new rule, shown as rule 3, and RuleNode 3 was created. Case 3 became the cornerstone case for RuleNode 3.
- Case 4 was presented to the SCRDR engine. After evaluating this case against the maturing rule tree, the last `TRUE` RuleNode for the case was RuleNode 3. The expert was dissatisfied with the conclusion from RuleNode 3. At that time in the IT support organization software patch 4.5.6 had become available for product A running on the windows platform with software versions greater than 1.0 and less than 2.0. The expert offered a new rule, shown as rule 4, and RuleNode 4 was created. Case 4 became the cornerstone case for RuleNode 4.
- Case 5 was presented to the SCRDR engine. After evaluating this case against the maturing rule tree, the last `TRUE` RuleNode for the case was RuleNode 3. RuleNode 4 was not satisfied. The expert disagreed with the conclusion from RuleNode 3. The expert offered a new rule, shown as rule 5, and RuleNode 5 was created. Case 5 became the cornerstone case for RuleNode 5.

In the SCRDR example given there were several instances where correct conclusions were rejected in the training of the SCRDR rule tree to make way for more specific conclusions. In fact, some of the conclusions provided applied concurrently, for example at RuleNode 3 three conclusions were in fact appropriate: 'Product A', 'Windows Platform', and 'Old Software'. This highlights the disadvantage of the SCRDR approach when trying to model real-world problems in which multiple classifications apply.

Note that the depth versus breadth of the SCRDR decision tree depends on the degree of overlapping classifications e.g. subsumption or exception style classifications in the chosen problem domain, as well as the order in which the cases and corresponding classifications are presented.

4.3.1.1 SCRDR in Pathology: PEIRS

The first major implementation of SCRDR was the Pathology Expert Interpretive Reporting System (PEIRS) that ran in routine operation over the four year period from March 1991 to

January 1995, interpreted thousands of cases, and grew to more than 2111 rules. PEIRS was used to add clinically meaningful interpretations and comments to pathology reports at the Department of Chemical Pathology at St. Vincents Hospital, Sydney (Kang, 1995, p 34). As well, PEIRS assisted with real-time decision support for test ordering by recommending follow-up tests as necessary (Edwards, 1996, pp 19, 157).

Edwards reports that PEIRS became one of the largest medical expert systems in routine use (Edwards, 1996, p i, 96). After four years of operation, PEIRS covered 12 sub-domains (p 101) and 800 (20%) of the 4000 reports per week received an interpretation. At 25 months of operation the accuracy was 97% over the domains covered with an average 116 interpretations per day (p 97).

PEIRS was implemented in a non-MS Windows computing paradigm on VAX VMS (Edwards, 1996, p 87). A separate PEIRS browser was written as a Hypercard stack for the Apple Macintosh to assist with browsing the knowledge base (p98).

Following a similar trend to that observed in the support centre context over the last 10-15 years, clinical pathology laboratories have experienced increasing volumes of cases, increasing complexity with new tests and new test methods, increasing expectations about the quality and appropriateness of interpretation and investigation, and ongoing demand for improved efficiencies in expenditure and turnaround times (Edwards, 1996, pp ii, 2, 31) (Compton et al. 2005).

In a similar fashion to help-desk support being offered direct to clients via self-help web-kiosks on the Internet and thus removing the technician from the customer interaction, there has been a trend in pathology towards near-patient point-of-care, removing the pathologist entirely from the request-test cycle (Edwards, 1996, p 31).

PEIRS proved the validity of SCRDR for automating interpretive reporting in clinical pathology (Edwards, 1996, p ii). There was a positive educative effect for laboratory scientists involved in report validation and some evidence that PEIRS improved the domain knowledge of pathology experts (p 128 - 130). There was also some evidence that PEIRS assisted with diagnosis and management decisions (p 130).

Edwards and Kang report that PEIRS drastically increased the quality of the output from the pathology laboratory, and reduced the load of experts since checking interpretations in a report was much easier for staff than composing and adding new interpretations (Edwards,

1996, p122, 215) (Kang, 1995 p 34). The improved quality and completeness of the laboratory database through better clinical data was seen to be a useful consequence of the PEIRS system (Edwards, 1996, p123). Note that consistency is seen to be a major contributor to quality as perceived by customers, hence the focus on minimising variance in various quality control programs (TQM[63], Deming[64], Six Sigma[65] etc).

The advantages of the RDR paradigm demonstrated by PEIRS appears to be two-fold:

1. RDR can make it simpler for experts to see the need for (i.e. to conceive of) and to construct new RuleNodes by constantly highlighting differences and similarities between the current case and cases previously interpreted by the KBS;
2. Specialisations of more general RuleNodes can be made incrementally, without having to re-codify the condition pathway that led to the parent RuleNode.

In contrast, the restriction to single interpretative comments by SCRDR was seen as the major limitation in PEIRS because it necessitated repetitious knowledge acquisition (Edwards, 1996, p 177). To paraphrase, in SCRDR multiple permutations of classifications resulted in a combinatorial explosion of concatenated conclusions (Edwards, 1996, pp iv, 133-135)[66]. For example, in the Arterial Blood Gas domain, there was at least a 7-fold duplication of KA (p135). Multiple Classification Ripple Down Rules (MCRDR) (Kang, Compton and Preston, 1995; Kang, 1995) made a significant contribution in resolving the repetitious KA problem and is described in section 4.3.2 on page 50.

4.3.1.2 Social Lessons from PEIRS

In PEIRS, pathology experts were obliged to carefully evaluate each interpretative comment, since there was known to be an important incidence of error. Experts validated all of the interpreted reports prior to dispatch and they were held accountable for any errors in the

[63] http://en.wikipedia.org/wiki/Total_Quality_Management

[64] http://en.wikipedia.org/wiki/W._Edwards_Deming

[65] http://en.wikipedia.org/wiki/Six_Sigma

[66] What is meant by this is that in SCRDR, `N` different conclusions could require up to `(N^2 -1)` separate RuleNodes. For example, conclusions A and B might be represented over 3 different RuleNodes in the system: A, B, and AB.

interpretations that they approved so that the responsibility for pathology outcomes rested fully with the pathologists themselves (Edwards, 1996, pp 91, 129, 140, 197, 215, 221). These factors were seen to be the primary reasons for PEIRS success and a new model for error management based on human expert validation was derived (pp 222, 224, 236).

Thus PEIRS acted as a corporate memory, remembering interpretations that had been made in previous contexts, and offering them up for re-use in repeated future contexts. Edwards felt that by participating in the feedback process, the clinician's viewpoint was valued and they were able to feel involved in, rather than alienated from the development and evolution of the decision support system (p213).

Critics of PEIRS felt that it would lead to an inevitable decline in clinical skills on behalf of the pathologists (Edwards, 1996, p132). Counter to this view, was the perceived positive educative effect (p 128 - 130).

PEIRS was accepted by pathologists because it permitted extensive use of locally derived protocols and customs, which were subject to frequent change (Edwards, 1996, p 165). Pathologists frequently disagreed openly with expert opinion stated in the medical literature, or by their clinical colleagues (p168). It was felt that it was highly unlikely that global and formal guidelines would ever replace locally derived protocols (p165), and that the reconciliation of local with global knowledge was perhaps the most challenging aspect of knowledge management (p226). These findings emphasise the need for large KBSs to represent multiple and parallel truths, and to provide a forum in which differences can be identified, highlighted, discussed and if appropriate, resolved.

4.3.1.3 SCRDR and Machine-Learning

Gaines and Compton (1995) used SCRDR as the underlying data structure for a machine learning algorithm known as *Induct* (see also Gaines, 1989). They report that its performance was similar to that of the well-known C4.5 machine-learning algorithm (Quinlan, 1993). As for other machine learning algorithms, background knowledge was required in the form of attributes and suitable selector functions to help distinguish the cases (p8).

When Induct was run on 9,514 cases from the GARVAN-ES1 data and compared with the manually acquired SCRDR database it was found that the total condition clause count for the manually acquired SCRDR system was 731, and for the automatically induced SCRDR system was 478. The 550 manually acquired rules had 1.3 clauses on average compared with

the 174 induced rules with 2.5 clauses per rule on average. Gaines and Compton (1995) deduced that in the SCRDR framework, experts tended to make a minimum amount of differentiation when adding new rules. This indicates that RDR may tend to create an over-generalisation bias in the knowledge acquired.

As well, SCRDR was used as the basis of C4.5 induction in the RDR-C4 system developed by Horn (1993, section 11). Horn notes the ability of inductive reasoning to suggest generalisations that human experts might otherwise have difficulty perceiving in a large knowledge base (sections 10 and 12). In section 13.5 (page 256) I describe how induction may be used to enhance KA in my proposed 7Cs system.

4.3.2 Multiple Classification Ripple Down Rules (MCRDR)

One of the limitations of SCRDR was that it only handled single classification domains. Where multiple classifications applied to an incoming case, compound conclusions were given that repeated the information provided for individual classifications. This was seen to exponentially increase the knowledge acquisition task since in numerous domains; many combinations and permutations of classifications are possible (Kang, 1995, p44). Given that most of the effort was in creating the interpretations in the first place, the repetition of interpretation content compromises the ease of system maintenance.

Multiple Classification Ripple Down Rules (MCRDR)[67] was developed to handle classification tasks where multiple independent classifications are required (Kang, Compton and Preston, 1995; Kang, 1996). In its original conception, MCRDR was implemented as a binary tree in which all pathways were evaluated except the true branch pathways of RuleNodes that failed to fire (Kang, 1995, p73). The system was soon modified to be an n-ary tree (Edwards, 1996, p 177).

The initial implementation was on an Apple Macintosh System 7 running Hypercard V2.1 with 4M RAM and 22Mbyte hard disk. The data file from the Garvan-ES1 domain took up about half the size of the hard disk, being some 10-12 Mbyte (Kang, 1995, MCRDR User's

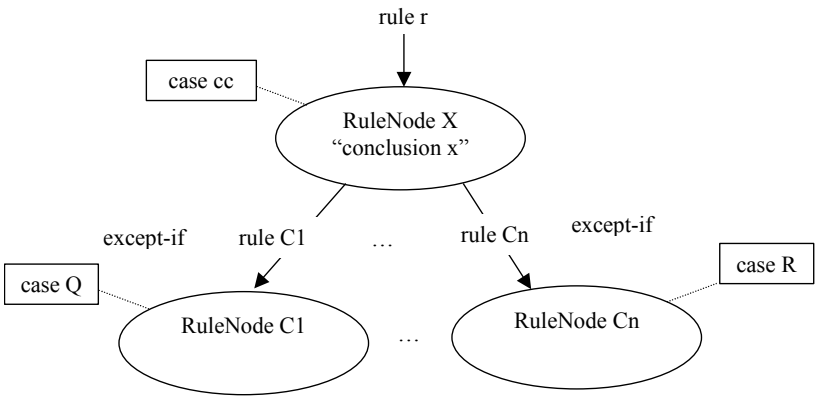
[67] A traditional MCRDR is formally a directed acyclic graph http://www.nist.gov/dads/HTML/directAcycGraph.html, as opposed to a decision tree. This is because some definitions of decision trees assume that only one attribute is considered at each node, rather than a set of conditions on multiple attributes. DAG is mentioned in (Dunham, 2003, p28).

Manual, p 1). MCRDR was developed at a time when the Internet was only just starting to impact workers outside of the ICT industry. Computer networking was still a luxury for many businesses, and computer-based collaboration was still a rarity. This history is important in so far as it would have limited the functionality conceived, and the capacity of system designers to implement it. These days, we are more limited by our imaginations than our ability to implement rich real-time interactive, collaborative and dynamic systems.

The remainder of this subsection details structural and algorithmic aspects of the Multiple Classification Ripple Down Rules (MCRDR) approach. For the sake of expediency, readers familiar with the technical aspects of MCRDR might like to skip forward to section 4.3.2.1 on page 57.

Figure 4 shows the nomenclature used to illustrate this system. An MCRDR can be defined as the set <*r,P,C,S,cc,x*>, where *r* is the rule for the RuleNode, *P* is the parent RuleNode, *C* is the child RuleNode, *S* is the next sibling RuleNode within the same level of decision list, *cc* is the cornerstone case for the RuleNode, and *x* is the conclusion.

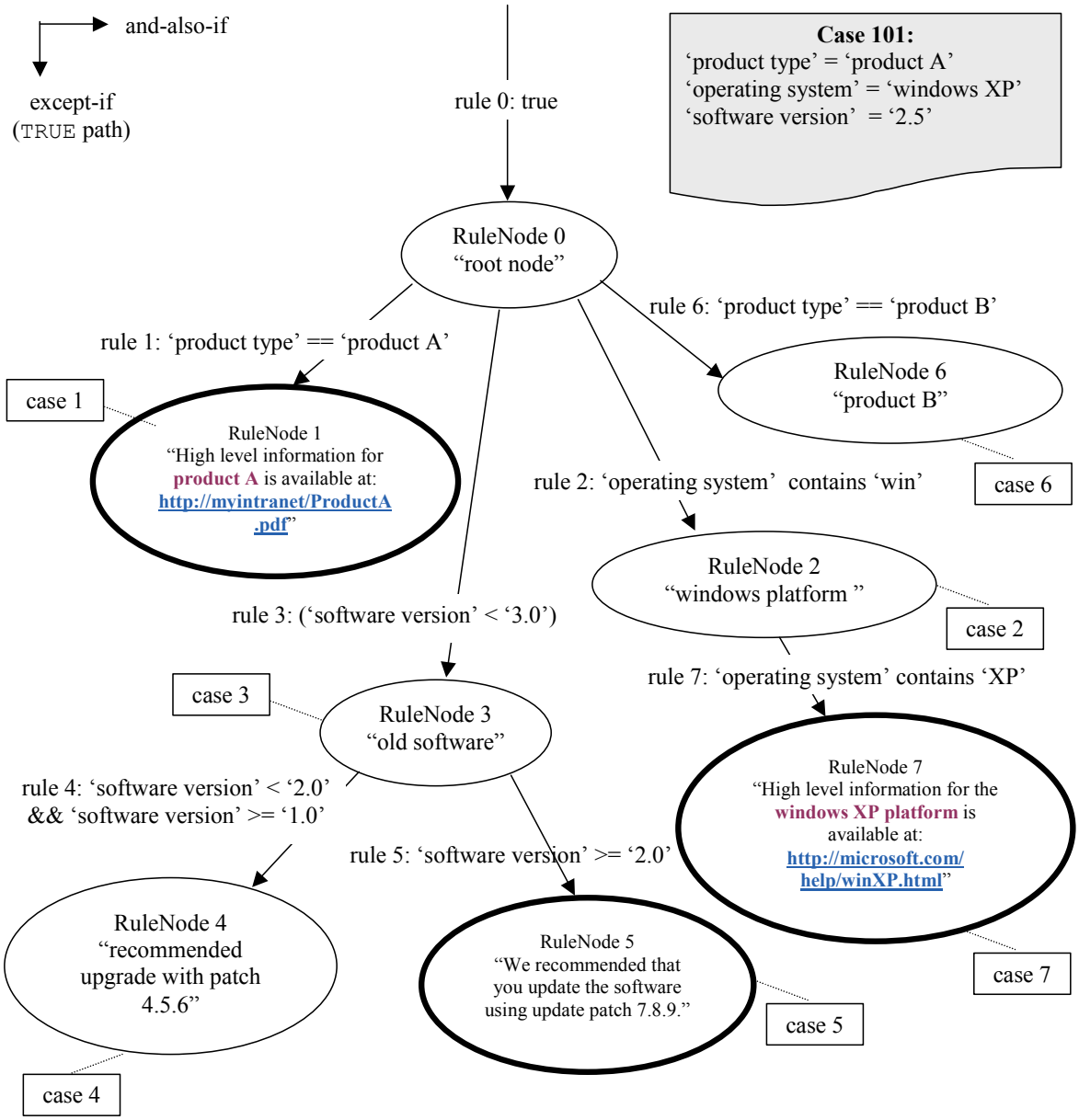
Figure 4: MCRDR RuleNode Nomenclature



In contrast to SCRDR RuleNodes can have multiple child RuleNodes attached to them. As for SCRDR, processing commences with the root RuleNode. As processing progresses, if a RuleNode evaluates to `TRUE` then all of its child RuleNodes are evaluated. If a RuleNode evaluates to `FALSE` then processing on that path through the rule tree stops. The last `TRUE` rule on each path through the rule tree constitutes the conclusions given.

The following shows an example MCRDR knowledge base.

Figure 5: **An MCRDR KBS**



Again, using Case 101 from our set of training cases in Figure 1 on page 42, the highlighted RuleNodes 1, 5 and 7 represent the last `TRUE` RuleNodes in each path through the rule tree, and hence they contain the final conclusions for Case 101. Cases 101 concurrently invokes the 3 emboldened classifications in Figure 5 at RuleNodes 1, 5, and 7. Using Case 101 as an example:

- Rule 0 evaluates to `TRUE` for Case 101 hence the case must be evaluated against all of RuleNode 0's child RuleNodes[68].
    - Rule 1 evaluates to `TRUE` for Case 101. RuleNode 1 has no child RuleNodes. Therefore the conclusion from RuleNode 1, namely "High level information for product A is available at: http://myintranet/ProductA.pdf" applies to Case 101.
    - Rule 2 evaluates to `TRUE` for Case 101. RuleNode 2 has a child node. Therefore Case 101 must be evaluated against that child node:
        - Rule 7 evaluates to `TRUE` for Case 101. RuleNode 7 has no child nodes. Therefore the conclusion from RuleNode 7, namely "High level information for the windows XP platform is available at: http://microsoft.com/help/winXP.html" applies to Case 101.
    - Rule 3 evaluates to `TRUE` for Case 101. RuleNode 3 has two child nodes. Therefore Case 101 must be evaluated against those child nodes:
        - Rule 4 evaluates to `FALSE` for Case 101. No further processing is required for RuleNode 4.
        - Rule 5 evaluates to `TRUE` for Case 101. RuleNode 5 has no child nodes. Therefore the conclusion from RuleNode 5, namely "We recommended that you update the software using update patch 7.8.9." applies to Case 101.
    - Rule 6 evaluates to `FALSE` for Case 101. No further processing is required for RuleNode 6.

So the multiple conclusions for Case 101 are that:

- High level information for product A is available at: http://myintranet/ProductA.pdf;
- High level information for the windows XP platform is available at: http://microsoft.com/help/winXP.html; and

[68] Conventional MCRDR systems commence with RuleNode 0 offering a NULL classification (Kang, 1995, pp 32, 60, 73).

- We recommended that you update the software using update patch 7.8.9.

In SCRDR only one cornerstone case is associated with each RuleNode. It is the case that first caused that RuleNode to be created.

In MCRDR however, in addition to the cornerstone case that caused that RuleNode to be created, there is a set of Cornerstone cases for a given RuleNode which is comprised of each case that caused either the RuleNode, or one of its dependent RuleNodes (e.g. its child, grand-child, great-grand-child etc) RuleNodes to first be created. For example, in Figure 5 the set of Cornerstone cases for RuleNode 0 would be {1,3,4,5,2,7,6}; the set of Cornerstone cases for RuleNode 3 would be {3,4,5}; and the set of Cornerstone cases for RuleNode 2 would be {2,7}.

In conventional MCRDR when the conclusion from a parent RuleNode is rejected, and a new alternative RuleNode is being constructed, the set of Cornerstone Cases for the parent RuleNode must be determined, and the new rule must distinguish the case on hand from each case in the Cornerstone set. The expert is presented with one cornerstone case at a time. The expert constructs a rule to distinguish the new case from the first case presented and then each case in the cornerstone list is evaluated to see if it is also distinguished by the new rule (Richards, 1998a, p55). If a case is satisfied by the rule the expert must add extra conditions to the rule to distinguish this case. This continues until all related cornerstone cases are distinguished.

The MCRDR knowledge base in Figure 5 was formed in the following manner:

- At first the only node in the MCRDR tree was the root node, RuleNode 0, with a rule that always returned `TRUE`, and a null or "root node" conclusion. All cases presented to the rule tree would satisfy this RuleNode.
- Case 1 was presented to the MCRDR engine. The expert disagreed with the conclusion from RuleNode 0. The expert offered a new rule, shown as rule 1, and RuleNode 1 was created. Case 1 became the cornerstone case for RuleNode 1 since it was the first case that caused RuleNode 1 to be created.
- Case 2 was presented to the MCRDR engine. The expert agreed with the conclusion from RuleNode 1, but the expert wanted to provide an additional conclusion. The expert offered a new rule, shown as rule 2 which differentiated Case 2 from the

Cornerstone Case set for RuleNode 0, namely {Case 1}, and RuleNode 2 was created. Case 2 became the cornerstone case for RuleNode 2 since it was the first case that caused RuleNode 2 to be created.

- Case 3 was presented to the MCRDR engine. The expert agreed with the conclusions from RuleNodes 1 and 2, but the expert wanted to provide an additional conclusion. The expert offered a new rule, shown as rule 3 which differentiated Case 3 from the Cornerstone Case set for RuleNode 0, namely {Cases 1,2}, and RuleNode 3 was created. Case 3 became the cornerstone case for RuleNode 3 since it was the first case that caused RuleNode 3 to be created.

- Case 4 was presented to the MCRDR engine. The expert agreed with the conclusions from RuleNodes 1 and 2, but the expert wanted to reject the conclusion provided at RuleNode 3 and provide an exception. At that time in the IT support organization software patch 4.5.6 had become available for software versions greater than 1.0 and less than 2.0. The expert offered a new rule, shown as rule 4 which differentiated Case 4 from the Cornerstone Case set for RuleNode 3, which at that time comprised only of {Case 3}, and RuleNode 4 was created. Case 4 became the cornerstone case for RuleNode 4 since it was the first case that caused RuleNode 4 to be created.

- Case 5 was presented to the MCRDR engine. The expert agreed with the conclusions from RuleNodes 1 and 2, but the expert wanted to reject the conclusion provided at RuleNode 3 and provide an exception. At that time in the IT support organization software patch 7.8.9 had become available for software versions greater than 2.0 and less than 3.0. The expert offered a new rule, shown as rule 5 which differentiated Case 5 from the Cornerstone Case set for RuleNode 3, which at that time comprised only of {Cases 3,4}, and RuleNode 5 was created. Case 5 became the cornerstone case for RuleNode 5 since it was the first case that caused RuleNode 5 to be created.

- Case 6 was presented to the MCRDR engine. The expert wanted to reject the conclusion provided at RuleNode 0 and provide an exception. The expert offered a new rule, shown as rule 6 which differentiated Case 6 from the Cornerstone Case set for RuleNode 0, which at that time comprised of {Cases 1,2,3,4,5}, and RuleNode 6 was created. Case 6 became the cornerstone case for RuleNode 6 since it was the first case that caused RuleNode 6 to be created.

- Case 7 was presented to the MCRDR engine. The expert agreed with the conclusions from RuleNode 1, but the expert wanted to reject the conclusion provided at RuleNode 2 and provide an exception. The expert offered a new rule, shown as rule 7 which differentiated Case 7 from the Cornerstone Case set for RuleNode 2, which at that time comprised only of {Case 2}, and RuleNode 7 was created. Case 7 became the cornerstone case for RuleNode 7 since it was the first case that caused RuleNode 7 to be created.

In the MCRDR system, if a new case is added to the system, the user can choose to accept a given conclusion, or alternatively reject the conclusion by creating a differentiating rule with an alternate conclusion. In that case, the following applies:

- The new rule must be a valid boolean expression, which the MCRDR engine can evaluate.
- The rule for the new RuleNode may optionally be restricted to a single test, e.g. that (‘movement’==‘flies’), rather than a conjunction of tests.
- The new RuleNode must have either a different conclusion, or a different rule compared to its sibling RuleNodes.
- The new RuleNode must test for some feature of the Review Case and must evaluate to `TRUE` for the Review Case.

In conventional MCRDR systems, new cases could be misclassified in one of three ways: one or more of the conclusions are incorrect, one or more conclusions are missing or a combination of incorrect and missing. The user could decide to stop an incorrect conclusion instead of replacing it with a new conclusion. In conventional MCRDR systems this was achieved by adding a stopping RuleNode[69] in the same way as adding other types of RuleNodes. The following table shows the three types of misclassifications dealt with in conventional MCRDR systems. As described in this table, the user may decide to stop an incorrect conclusion instead of replacing it with a new conclusion.

[69] In conventional MCRDR systems, a stopping RuleNode has a “stop” conclusion that prevents that RuleNode from being displayed for cases that satisfy the rule conditions of the RuleNode.

Table 6: The three ways in which new RuleNodes correct a knowledge base

Adapted from Kang (1995, p63).

Scenario Identity	Wrong Classification	Conclusion Status	To correct the KB
A: Reject	Wrong classification to be stopped	Incorrect	Add a stopping RuleNode at the end of the path to prevent the classification
B: Replace	Wrong classification replace by new classification	Incorrect and Missing	Add a RuleNode at the end of path to give the new classification
C: New	A new independent classification	Missing	Add a RuleNode at a higher level to give the new classification

The decision of where to add a new rule to the KB is affected by the design of the user interface, user preferences and the situation. If rules tend to be added to the top level the domain will be covered more rapidly but there may be greater over-generalisation and hence false-positive errors. If rules are added to the end of pathways there may be greater over-specialisation and hence false-negative errors, and hence the domain coverage will be slower (Kang, 1996).

Note that in MCRDR when a parent RuleNode is *live* for a given case, it is because the ancestor RuleNodes of that parent RuleNode all evaluate to `TRUE` for that case, and the child RuleNodes of that parent RuleNode all evaluate to `FALSE` for that case. Hence the condition path for that case is equal to the conjunction of the conditions of each of the ancestor RuleNodes and the negation of the conditions of each of the child RuleNodes of the given RuleNode.

4.3.2.1 MCRDR and Pathology: PEIRS

As part of his PhD thesis, Edwards ran trials on an MCRDR engine using the PEIRS data (Edwards, 1996, p 193). He found that while KA took between 30 seconds and 6 minutes in the MCRDR system as compared with 30 seconds to three minutes for the SCRDR system, the overall time for knowledge base construction was 210 minutes for MCRDR as compared with 270 minutes in the SCRDR system. (In this trial, the new improved TCRDR user interface was used for the SCRDR system). At the conclusion of the comparative KA period he found that far fewer KA incidents were required in MCRDR (70 of 262 cases were incorrect i.e. 27%) as compared with SCRDR (180 of 262 cases were incorrect i.e. 69%).

Hence the faster maturation and greater accuracy of MCRDR for multiple classification domains as compared with SCRDR was confirmed.

4.3.2.2 MCRDR and Pathology: Pacific Knowledge Systems

Pacific Knowledge Systems (PKS) was founded in 1996 and has found success in commercialising MCRDR in the pathology domain. Their LabWizard KBS assists with laboratory disease diagnosis, for example using blood and urine samples (see http://www.pks.com.au). Please refer to Appendix G (commencing on p 408) for the transcript of an interview that I arranged with PKS in June 2005.

Compton et al. have also recently reviewed the MCRDR experience at PKS in the chemical pathology domain (Compton at. al. 2006). They report that at one laboratory over a 29 month period over 16,000 rules were added and 6,000,000 cases were interpreted by PKS's LabWizard MCRDR software product. This particular lab maintains about 19 separate knowledge bases, and generates in the order of 300,000 patient reports per month with 80% of reports being auto-validated after a 5-month introduction period (presumably using prudence techniques as outlined in section 13.8 on page 263), and with a modification rate on generated reports of less than 1.5%.

Laboratories that use LabWizard tend to develop multiple concurrent but separate knowledge bases for different sub-domains managed by different experts. Currently LabWizard does not offer support for multiple experts to resolve their classification conflicts within the scope of an MCRDR interaction.

U.S. Patent No. 6,553,361 (issued to Compton, et al. on 22 April 2003) was created with the input of PKS staff to provide intellectual property (IP) protection for their LabWizard software. The patented system allowed only one expert to update the system at a time. Cases, and particularly cornerstone cases were static – they mostly contained results from blood and tissue samples, or soil samples, and were therefore unchanging. The focus on the system was mostly on generating high volumes of easy to read diagnostic pathology lab reports.

4.3.2.3 MCRDR and the Help Desk

MCRDR has been previously explored in the help desk environment (Kim, 2003) and Kang et al. (1996). In her PhD thesis, Kim (2003) applied the concept lattice from Formal Concept Analysis (FCA) (Wille, 1992) to generate a browsing structure to assist users in navigating

the knowledge base. These MCRDR help desk implementations were limited to KBS updates by a single expert, and the use of simple keyword-based attributes and rules.

The prototype described by Kang et al. (1996) combined a keyword search with Case-Based Reasoning indexing techniques to provide a guided MCRDR interaction that was able to quickly steer users to appropriate help information on the Internet[70]. As noted by Kang et al. (1996) the MCRDR engine has two problems as an information retrieval engine. The first one is the number of conditions that are to be reviewed by the user. The second one is the number of interactions between the user and the system. The prototype by Kang et al. (1996) attempted to minimise this problem by allowing users to apply a keyword search to effectively pre-filter the rule tree to only include those cases that satisfied the keyword search criteria. The user could then interact with a minimised MCRDR rule tree to select the relevant conclusions and update the knowledge accordingly. The keyword search data effectively became the attribute-value (A-V) data for the search case.

The prototype described by Kim, Compton, and Kang (1999) extended the Kang et al. (1996) prototype by allowing an expert user to also build and maintain the help desk document knowledge base by applying keywords to help documents[71].

In contrast to previous MCRDR approaches, including previous approaches to build a system for the help-desk domain using MCRDR, in this research it was found that:

1. Experts in the support centre domain have to define the cases, as well as the rules, and
2. Input is required from multiple experts to provide the necessary domain coverage.

In this context, what is meant by *defining the cases* is that experts need to identify the relevant features for cases in the domain, that is, the structure of the domain model, the questions that help desk personnel might ask the customer in identifying the nature of the problem, and the

[70] Some of this text appears in (Vazey and Richards, 2004).

[71] More recently Park, Kim and Kang (2003, 2004) have been involved with web monitoring that also uses keyword and MCRDR to organise document access.

conditions under which these questions should be asked (including the order of the presented questions)[72].

4.3.2.4 MCRDR and Security Information Analysis

The use of RDR has been promoted for security information analysis in (Yao et al., 2005). That review demonstrates that the RDR approach provides a mechanism for simultaneously acquiring both a subsumption and an exception hierarchy. The subsumption hierarchy is one that will be familiar to people that work with ontologies, taxonomies or object oriented (OO) modelling tools and languages. It represents the is-a relationship between a child class and its parent e.g. *a penguin is-a bird*. In contrast, the exception hierarchy allows simple heuristics (rules of thumb) to be gathered that help users quickly classify and derive conclusions for cases in a domain by providing exceptions to more general rules. For example: *birds fly, except in the case of penguins, emus and ostriches*. It corresponds to polymorphism in Object Oriented (OO) modelling.

4.4 Variations on the RDR theme

Over the past decade there have been numerous implementations of SCRDR and MCRDR[73]. In general, cases are presented to an expert who accepts a conclusion or creates a new rule with the correct conclusion that is attached to the incorrect rule. An approach to generalising RDR has been provided by (Compton, Cao and Kerr, 2004). Other RDR variations related to this research are described below. Later on in the thesis (section 6.7, page 87) further discussion is provided of some of these RDR variations.

4.4.1 H-RDR

Heuristic RDR (H-RDR) was proposed by Bekmann (et. al 2004) (Bekmann, 2006). The approach combines incremental knowledge acquisition and probabilistic search algorithms, such as evolutionary algorithms, to allow a human to develop problem-solvers in new

[72] Many thanks to an anonymous PKAW 2006 reviewer of the (Vazey and Richards, 2006b) paper for some of these very helpful reflections on the research presented. The importance of controlling the order of rule evaluation is also noted in (Compton, Cao and Kerr, 2004).

[73] Implementations of the RDR variations are noted for example in (Kang, 1995, pp 40, 41), (Richards, 1998a, p67), and (Compton, Cao and Kerr, 2004, p2).

domains. The HRDR framework is also referred to as HeurEAKA. It has recently been demonstrated in traffic light optimization, and the optimization of channel routes in VLSI design. Genetic algorithms were applied together with a Bayesian optimization algorithm (Bekmann, 2006, p 30).

4.4.2 R-RDR

Kang (1995, p132-133) reports on the use of SCRDR for ion chromatography in which an inference mechanism was developed that automatically filled in missing values to assist with the configuration of the ion chromatography equipment (Mulholland et. al, 1993). Each sub-domain was represented in a separate single classification knowledge base, which the inference engine would bring together to reason about and hence configure the case at hand (personal communication, Richards, 2006). The mechanism became known as Recursive RDR (R-RDR). On each cycle the last true conclusions at the end of each path were returned. A heuristic was then used to choose which conclusions to accept. These were used as inputs for the next inference cycle.

4.4.3 P-RDR

Possible RDR (P-RDR) considers a subset of the branches used in RDR and reports all last true conditions after taking any branches that might possibly be true. One implementation, which was an initial and partial solution to the ion chromatography configuration problem, combined RRDR with Possible RDR (PRDR) (Mulholland et al 1993).

PRDR can be difficult to manage as often too many alternatives are provided and it is hard to determine the useful ones. In PEIRS, it was noted that uncertainty was sufficiently represented in the wording of conclusions, without requiring any further representation in the underlying RDR structure (Edwards, 1996, pp 120-121).

4.4.4 I-RDR

Interactive RDR (I-RDR) was previously undertaken for single classification tasks, but the ideas were not published[74]. In concept, I-RDR allowed an SCRDR system to prompt the user for more information when required. The user was able to enter a value or indicate that the value was "unavailable" and the response was added to the case (personal communication,

[74] Personal communication, Quoc Thong Le Gia via Debbie Richards (2004).

Debbie Richards, 2006). When changes were made to the temporary case, no matter how small, another path could be found through the decision tree when case was re-evaluated. The cases were temporary and were never stored. Such cases were simply used as a way of querying the system in a directed way. Simulation studies were done using the IRDR mechanism that yielded poor results and the idea was not pursued.

4.4.5 N-RDR

Nested RDR (N-RDR) introduced the concept of intermediate conclusions (Beydoun and Hoffman, 1997). The lack of tools for abstracting specific features to more general features was seen as an important limitation in PEIRS (Edwards, 1996, pp 120, 134, 137) and N-RDR suggested a solution to this problem. N-RDR allowed knowledge at different levels of abstraction to be captured in separate knowledge bases that the inference engine would bring together to reason about the case at hand (personal communication, Richards, 2006). Drake and Beydoun (2000, p76) extended N-RDR to allow the expert to include parameter lists for expert-introduced concepts.

NRDR is suited to single classification problem domains. However, as noted previously in section 2.5.3 (page 26), the troubleshooting domain in the HTG support centre case study was a multiple classification domain. In addition to the changes proposed later in this thesis to adapt MCRDR to support collaboration, the shared child RuleNode structure presented in section 13.4 (page 245) and the use of the setAttribute() conclusion type (discussed on page 195) allows the multiple hierachical restricted domains (MHRD) supported by NRDR (Beydoun et. al, March 2005) to be supported using the 7Cs (modified MCRDR) approach proposed by this research.

4.4.6 TC-RDR

Time Course RDR (TC-RDR) was developed by Philip Byrnes-Preston to investigate the interpretation of "time course" (temporal) data (Edwards, 1996, p144). One of the main innovations introduced by TCRDR was the ability for users to create and edit functions, providing for high-level feature abstractions. The feature was utilised extensively in PEIRS (Edwards, 1996, p154).

4.4.7 Rated MCRDR

Rated MCRDR was proposed by Dazeley and Kang (2004) and offers a way of reducing the brittleness of a KBS by identifying when a given case follows an unusual pattern of paths through the decision tree, and therefore has a higher risk of interpretative error and the subsequent need for knowledge acquisition. It uses an MCRDR front-end and a neural network back-end that provides a numeric confidence rating for combinations of classifications. "*When an unusual classification pattern was found then the system provides a warning to bring the case to the user's attention*" (Dazeley and Kang, 2004, p11).

4.5 Chapter Summary

The main ideas embodied in SCRDR and MCRDR are: KBSs grounded in cases, an exception structure, and a simple KA technique designed to be performed by the domain expert that encourages the incremental development, maintenance and validation of the knowledge base.

MCRDR offers an algorithm and data structure for indexing classifications by creating rule conditions that examine the attributes of incoming cases. Hence MCRDR could be used to develop an index for support centre solutions i.e. solution classes, on the basis of incoming problem cases. The case-driven approach to acquiring knowledge employed by MCRDR could work well in the support centre environment where users continuously pull cases off an incoming problem queue and attempt to match them to solutions.

However, in contrast to previous MCRDR approaches, including previous approaches to build a system for the help-desk domain using MCRDR, in this research it was found that:

1. Experts in the support centre domain have to define the cases, as well as the rules, and
2. Input is required from multiple experts to provide the necessary domain coverage.

The next chapter discusses trends in collaborative knowledge acquisition that are relevant to the trouble-shooting context.

CHAPTER 5: COLLABORATION FEATURES AND TRENDS

5.1 Chapter Outline

Recalling from section 1.3 on page 3 that:

> *a sub-goal of this research was to explore the possibilities to adapt SCRDR and / or MCRDR to the support centre context, and to design, prototype and test a software blueprint that could assist with problem solving in previously unchartered, repetitious and complex problem domains;*

as described previously (section 3.5.1 on page 36) the HTG interview and survey results showed that trouble-shooting knowledge is learned, acquired, generated and consumed in a decentralized manner by multiple stakeholders and using multiple sources of information across the global support organization. As a result of previous software development, project management and consulting experiences[75], I came to the view that RDR needed to support collaboration. Related ideas were presented to HTG in March 2004, and published in (Vazey and Richards 2004a) (Richards and Vazey, 2004) (Vazey and Richards, 2004b) and (Vazey

[75] At one of the firms (for which I was a Senior Design Engineer from 1993 to 1998) we used Rational Rose's (ne. PureAtria's) Clearcase collaborative software version control system to manage massive multi-user collaborative software development projects across multiple sites in multiple different countries. At another firm (for which I was a Project Manager from 1998 to 2000), we used the GNU open-source CVS product to support multi-site multi-user software collaboration. Both source control solutions had the ability to compare and merge shared fragments of source code that were concurrently modified by numerous individuals in the software development team. There was a separation between private user views, and the public system view. Similarly products like DDTS and Lotus Notes supported international collaboration in requirements management and defect tracking across multiple and varied sites, cultures and timeframes. At another firm (for which I was a Senior Software Consultant from 2000 to 2001), a multi-user heavily loaded RAM-database client-server Internet solution provided for real-time transaction locked database updates by volumes of online concurrent users using (amongst other patterns) the model-view-controller design pattern (Burbeck, 1987). Finally, (while I was consulting) at yet another firm in the ICT industry (2001), collaborative project management was achieved using the distributed web-based MS Project platform, and collaboration in requirements engineering, design, and testing occurred using Wiki's OpenSource software. Combined with my experiences in botanical classification as outlined on page xi, I found compelling reasons to pursue a collaborative classification approach to knowledge acquisition.

and Richards, 2005a)[76]. The collaborative RDR ideas later formed the basis of a PCT patent application as described on page 283.

Since no one person could possibly hold all the answers to incoming problems at the support centre, perhaps the most significant limitation with existing SCRDR and MCRDR approaches at the time of this research was their inability to support multiple experts in collaboratively constructing the knowledge. As noted by Dr. James Moody in his keynote speech at the very recent Macquarie University Innovation Awards (Nov 29, 2006), the ICT revolution has had a clear focus on reducing the transaction costs between geographically dispersed people and organisations. In the last 15 years a plethora of collaborative software products have therefore emerged.

In the SCRDR-based PEIRS system, all reports that were misinterpreted by the system were given to a single expert to add new rules to the system (Kang, 1995, p39). Further, as described in section G.10 (page 411), at the time of this research only one person at a time could build the rules in PKS's MCRDR-based LabWizard product. There was no mechanism for multiple people to simultaneously add rules to the same knowledge base. Each sub-domain was codified and maintained in a discrete single-user knowledge base. The prevailing single-expert paradigm at the time of this research was also reflected in the definition of a KBS by (Compton, Cao and Kerr, 2004, p2) as a system in which rules "*are added to capture the preferences and beliefs of the owner/supervisor/teacher of the system in some sort of knowledge acquisition process*"[77]. As discussed previously (section 3.4 on page 34), the lack of support for collaboration was shared by state-of-the-art ontology KBSs. Hence the problem of collaborative MCRDR knowledge base construction was one of the main limitations addressed by this research.

This chapter addresses the following research question:

Q3. What literature and technologies are relevant to a collaborative MCRDR-based troubleshooting approach?

[76] A list of publications arising from this research is provided on page 281. Many thanks to Debbie Richards for much appreciated work on the co-authored papers.

[77] It is interesting to note that further on in (Compton, Cao and Kerr, 2004, p3) "*the philosophical arguments for why an (individual) expert can never be relied on*" by (Compton and Edwards, 1990) are referenced.

Findings from CSIRO's Panoptic search engine project are presented (section 5.3, page 71), showing that anchor text – the text that webmasters use to label their links to others' websites – provides by far the best correlation with search terms that people use when attempting to locate websites using popular search engines like Google.

New age collaborative web platforms like blog sites, web forums, and folksomonies demonstrate the enormous will and capacity of users from disparate walks of life to communicate and collaborate. Wikipedia is an outstanding example of this phenomenon. In this chapter these technologies are described, and the relationship between folksomonies, Wikipedia and the anchor text phenomenon harnessed by popular search engines is highlighted.

The anchor text analogy is used to suggest the collaborative use of rule-based anchor text to tag problem classes for the support centre, and thereby to provide an index from problem cases to their relevant solutions.

5.2 Collaboration and Conflict

As mentioned previously, at the time of this research, previous RDR systems had only ever supported one person at a time updating the RuleNodes in the rule tree. Although work had been done in trying to resolve conflicting knowledge models by comparing Formal Concept Analysis (FCA) (Wille, 1992) and (Ganter and Wille, 1999) representations of classifications and rules collected using the MCRDR approach (Richards, 1998a, Chapter 6, pp 181 – 209), there had been no support for multiple experts to collaboratively resolve their classification or conclusion conflicts inside the scope of a set of SCRDR or MCRDR interactions. No mechanisms had been developed to support the inline conflict resolution required when acquiring knowledge from multiple and possibly conflicting sources of expertise.

This thesis supports collaboration by multiple experts in concurrently acquiring and negotiating knowledge representations in a shared domain. In contrast, and similar to the approach taken by Richards (1998a) during the course of this thesis, Beydoun et. al (March 2005) studied the merging of separately created NRDR KBSs from independently operating experts that modelled identical domains. They refer to their work as *cooperative*, as opposed to *collaborative*. A discussion of the differences between these two terms was provided in section 1.5 (page 5). In support of this thesis, (Beydoun et. al, 2005, p48) argue that '*a coherent collective experience is a better reflection of "reality*"' than the experience of

disparate individuals and that "*a state of consensus between experts reflects a mature model, which is consistent with the world as perceived by the collection of these experts*"

In separate cooperative work, virtual surfing trails are captured from multiple web surfers in a given topic area that can be shared by peers in that topic area using FCA (Beydoun et. al, 2007). In that paper, they argue that "*it is of high interest to learn from colleagues of the same community because of common interests and aims*" (Beydoun et. al, 2007, p3)[78].

Almost a generation ago, Shaw and Gaines articulated the importance of seeking input from multiple experts when attempting to elicit knowledge about a domain (1989, p1):

> "*in many domains… expertise is… distributed over many experts, and the purpose of the system is to bring it together… the use of multiple-experts… is an attractive technique to prevent individual experts…(from) failing to fully explore and express their conceptual domains".*

As well, they note that (p21):

> "*in the initial phases of knowledge acquisition, highlighting gross similarities and differences is itself valuable in promoting directed discussion among experts… that can lead to the exposure of more subtle relationships*"

Easterbrook (1991, pp 6, 11) highlights that conflict in both interpretations (inputs) and goals (outputs) is an inevitable feature of group interaction, knowledge elicitation, and system design; and it has a useful role in facilitating change and producing higher quality group decisions as it involves questioning and evaluating received wisdom. He highlights that avoidance is only one strategy for dealing with conflict (pp 1-2) and he suggests that (p13):

> "*The first problem for conflict resolution is to recognise that a conflict exists*".

Conflict is also discussed by Zondag and Lodder (2005, p5) who note that goal incongruity is mostly of a perceived nature, and that any information exchange to improve the level of objective and reliable information that each party holds about the other party's goals is likely

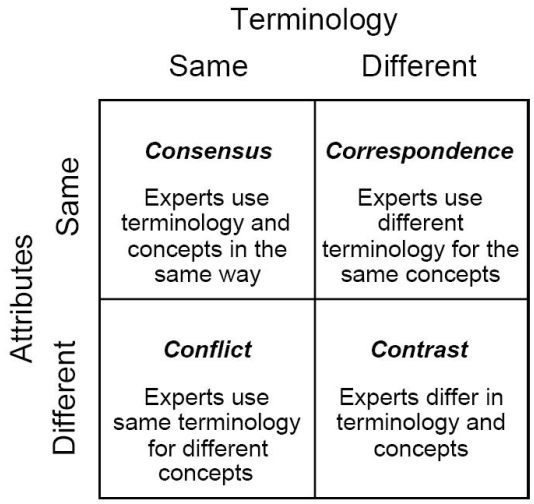
[78] Note that with reasonable computational delays in mind, an FCA approach to knowledge viewing could be applied at any time in the 7Cs approach proposed by this research, for either the public knowledge view shared by the expert group, or the private knowledge views of individual experts.

to reduce the conflict. Further, examples of terminological and interpretational conflict are highlighted by Hakimpour et al. (2001).

The analysis by Shaw and Gaines (1989) highlights that consensual agreement is only one of many significant outcomes from the involvement of multiple experts since experts may legitimately have different terminologies for the same domain concept; they may be operating out of different conceptual frameworks; they may be operating from different perspectives within the same conceptual framework; and they may choose to describe things at different levels of abstraction.

Figure 6 reproduced from Shaw and Gaines (1989, p3) shows the 4 different relations that can result from the different distinctions (attributes) and terms used by two or more different experts operating in the same domain, namely: consensus, correspondence, conflict, and contrast.

Figure 6: Consensus, conflict, correspondence, and contrast among experts



Reproduced from Shaw and Gaines (1989, p3)
with the kind permission of Brian Gaines.

As a result of these different terminologies and distinctions, Shaw and Gaines argue that support for the coexistence of a variety of corresponding concepts will increase the usability of the expert system (1989, p5).

Easterbrook (1991, pp 12, 13) notes that a good conflict resolution approach will emphasise an appropriate level of communication between the different parties: increased communication leads to decreased conflict up to a certain level, but too much communication

can lead to increased conflict, therefore a balance must be struck between encouraging communication and devoting appropriate amounts of effort to resolving the differences. It is only worthwhile entering the conflict resolution process when the differences between viewpoints matter. He goes on to develop a scale for the severity of the conflict ranging from *non-interference* where merging or co-existence provides a simple solution, to *mutually exclusive* where the two different viewpoints cannot continue to co-exist and must be resolved (pp 21, 22). In the system proposed further on in this research, the shared child RuleNode structure (described in section 13.4 on page 245), supports the co-existence of multiple concurrent truths demanded by multiple experts operating within the same KBS.

As well, two users may wish to view conclusions at a RuleNode at different levels of abstraction. This issue is highlighted by Golder and Huberman (2005) in reference to collaborative tagging systems (described later in section 5.4 on page 72), where users perform a significant amount of tagging for personal use, rather than public benefit, and where users display very different desires for specificity versus generality in the tags that they apply. In the PEIRS system different categories of comment or conclusion were required relating to pattern identification, clinical abstraction, and advice (Edwards, 1996, pp 77 – 78). In my proposed system, maintaining multiple conclusions for each RuleNude, and allowing the user to configure the types or levels of conclusions they wish to see in their output view should help to resolve this problem. We return to this issue in the example provided in Appendix R (page 485).

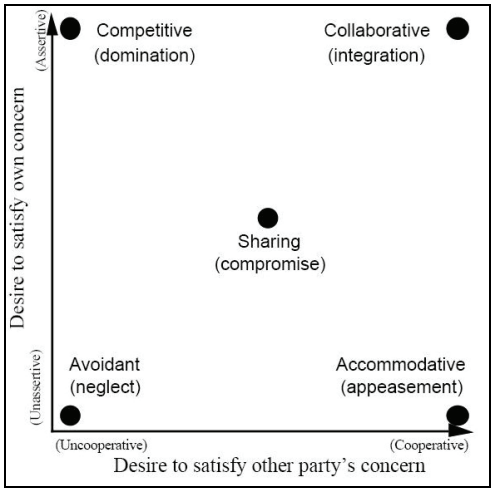
When multiple experts are involved, the problem of redundancy and duplication is particularly likely to arise in conventional MCRDR systems that provide rules with only conjunctions (ANDs) rather than ORs (Kang, 1995, p66) since alternate pathways to the same conclusion can only be represented by physically separate pathways. In contrast, the use of ORs in the rule conditions, together with the use of the shared child RuleNode structure in the system proposed by this research, will alleviate this problem.

Where the conflict is important enough not to be avoided, Easterbrook (1991) advocates that a project will be enhanced by handling the conflict in more direct ways. In the system proposed by this research, the tracking of live versus registered case-RuleNode associations (described in section 11.7 on page 210) supports the conflict resolution required when either partially-interfering or mutually exclusive viewpoints arise from multiple experts operating within the

same KBS. Similar to ideas subsequently presented in (Beydoun et. al, 2005, pp 66, 68) the '*integrated (i.e. "live") model… comes to represent a consensus between models provided (i.e. "registered") by different experts*'. Both works are based on the idea that "*internal inconsistencies in a cooperative modelling process are mainly a signal of incompleteness of the evolved model*".

Easterbrook (1991, p4) refers to three broad types of conflict resolution identified by Strauss (1978): collaborative (co-operative) methods that include education and negotiation; competitive methods that include combat, coercion, and competition; and third party methods that include arbitration and appeals to authority. In Figure 7, Easterbrook adds two other broad types of conflict resolution identified by Thomas (1976): sharing (compromise) and avoidance (neglect). The 5 different types of conflict resolution are plotted according to the desire to satisfy another party's concern, and the desire to satisfy one's own concern.

Figure 7: Behavioural modes of tackling conflict with (in brackets) the outcome sought in each mode.



Reproduced from Easterbrook (1991, p6)
with the kind permission of Steve Easterbrook.

From Figure 7 we can see that in a collaborative conflict resolution approach, participants seek to understand their differences and achieve a mutually beneficial solution (Easterbrook, 1991, p6). It is appropriate where participant's insights and commitment are important and need to be merged rather than compromised. The educative component of collaborative conflict resolution allows experts to gain a better understanding of the problem, better

understand each-other's viewpoints, and possibly reformulate the problem so that it disappears or becomes unimportant (Easterbrook, 1991, p4).

Easterbrook (1991, p3) highlights some of the problems of conflict avoidance: conflicts get suppressed; a single perspective gets adopted at the cost of alternative perspectives; the suppression eventually leads to dissatisfaction and may result in the withdrawal of participants; and the participants will maintain different understandings which may cause ongoing confusion. If resolution of the conflict occurs it will occur outside the framework of the method and will therefore be untraceable, making important rationales undetectable, and important decisions irreproducible.

5.3 Collaboratively Generated Anchor Text

Panoptic[79] is an enterprise search appliance, developed by CSIRO and the Australian National University (ANU), designed to dramatically improve the effectiveness of information retrieval for an organisation's customers and staff. In his "Enterprise Search and Metadata" presentation at the Macquarie University HCSNet Summer School (Dec 13-14, 2005, Sydney), David Hawking from the CSIRO provided some important insights into the search algorithms used by popular search engines such as Google, Yahoo and MSN.

Hawking described how the Panoptic project found that the most valuable information to popular web search engines and enterprise search appliances alike, was the *anchor text* information applied by other independent webmasters on the Internet to refer to a given website.

For example in HTML the anchor reference:

Look At Me

applies the anchor text: *Look At Me* to refer to the website shown.

It was Amatay (1998, p3) who originally noted the importance of anchor text to Internet searching. Amatay found that the anchor text used by others to refer to a given web page has a very strong correlation with the search text users apply when searching for that web page. According to David Hawking (personal communication, 2005) Panoptic's findings

[79] http://www.csiro.au/csiro/content/standard/pps6f,,.html

corroborate with Steve Robertson's findings[80] (then at City University, London). Robertson found that as far an anchor text is concerned: less common query words are more discriminating[81]; the more occurrences of a query word the better; and that shorter documents are preferred by users and should therefore be preferred by the search algorithm. Further findings from CSIRO's Panoptic project regarding the importance of anchor text to information retrieval algorithms is presented in Appendix H (commencing on p 414).

5.4 Folksomonies

A collective attempt to solve the findability[82] problem (presented by web forums, blogs[83] and the Internet in general) and to simulate the anchortext phenomena (discovered by Amatay, and harnessed by Panoptic and others), without requiring users to create their own web pages, has been the folksomony[84], also known as collaborative tagging or tagsomony.

Collaborative Tagging describes the process by which many users add metadata in the form of keywords (i.e. anchor-text) to shared content. Traditionally such categorising or indexing was performed by some authority, such as a librarian, or else derived from the material provided by document authors. Collaborative tagging is well suited to situations where either there is nobody in the librarian role or there is simply too much content for a single authority to classify (such as on the web, and in the support centre trouble-shooting context), and there is no way of deriving suitable classification indexes from the material itself (Golder and Huberman, 2005).

Folksomonies or tagsomonies offer an attempt to improve the findability of websites, including blog and web-forum entries. The term "folksomony"[85] was first coined by Thomas

[80] leading to the Okapi BM25 formula

[81] This also correlates with the Inverse Document Frequency (IDF) measure mentioned in Appendix D in relation to information retrieval.

[82] It has been estimated that the world's total store of knowledge is doubling every four years or less. Today's problem is not so much one of searchability, but rather findability[82], a term coined by Peter Morville (2005).

[83] for a good blog definition see: http://help.blogger.com/bin/answer.py?answer=36&topic=16

[84] http://en.wikipedia.org/wiki/Folksonomy

[85] http://vanderwal.net/essays/051130/folksonomy.pdf

Vander Wal[86] (2005) to describe forums in which people can tag objects (web pages, photos, videos, podcasts, etc., essentially anything that is Internet addressable) using their own vocabulary so that it is easy for them to re-find that information again. Tags are a form of keyword metadata (i.e. anchor-text), designed to help make Internet content more findable.

Vander Wal describes folksonomies as a social phenomena, since at least for broad[87] folksomonies, others that use the same vocabulary are able to find the object as well. He offers *del.icio.us* and *flickr* as examples of broad versus narrow folksomonies and notes that folksonomies work wonderfully for individuals trying to re-find their own information, but they work best for groups of users when the tags used to describe objects are part of a common vocabulary. Other popular examples of folksomonies include *furl*[88] and *shadows*[89]. A brief review of *del.icio.us* and *flickr* is provided in Appendix I (commencing on p 415).

Clay Shirky[90] (2003) explains how the freedom of choice for users forms a power law distribution for both tag popularity in broad folksomonies as well as web page popularity in general, that fits the 80/20 rule and a power law probability distribution[91] first identified by economist Vilfredo Pareto (1906) in regard to wealth distribution. Further discussion of this phenomenon is provided in section 7.2.6 (page 118).

Based on the success of the folksomony paradigm and the findings of a number of researchers including CSIRO's Panoptic project discussed earlier, it seemed plausible that a knowledge management system that relies on some form of article tagging by multiple users could be created to increase the findability of knowledge articles in the system for both individuals, and groups. These tags need not be restricted to simple keywords, but rather they could contain more complex expressions of rule conditions satisfied by the items being tagged, such as the

[86] http://vanderwal.net/

[87] for a discussion of broad (multi-user, multi-tag) versus narrow (single-user and/or single-tag) folksonomies see http://www.personalinfocloud.com/2005/02/explaining_and_.html

[88] http://furl.net/

[89] http://shadows.com

[90] http://www.shirky.com/writings/powerlaw_weblog.html

[91] http://en.wikipedia.org/wiki/Pareto_distribution

rule conditions in RuleNodes that are used to tag classifications in an MCRDR-based data structure.

Table 7 on page 74 summarises the analogy that this research draws between Folksomonies, Anchor text, and a Collaborative MCRDR approach, which in the tradition of RDR variations is referred to here as C-MCRDR.

Table 7: Analogy between Folksomonies, Anchor text, and C-MCRDR

Approach	**Who does the tagging?**	**Tag with what?**	**Why?**	**Search with what?**
Folksomony	Folksomony members.	Natural Language or Plain text Keywords	These keywords provide a match with the keywords or natural language that searchers use to locate the selected Internet resource e.g. website.	Keywords or natural language search text.
Anchor Text	Webmasters across the Internet community.			
Collaborative MCRDR (C-MCRDR)	Members of the collaborative MCRDR community.	Rule Conditions	These rule conditions provide a match on the semi-structured case data that searchers use to locate the selected Internet resource e.g. website.	Semi-structured cases.

5.5 Wikipedia

Wikipedia[92] is a collaborative knowledge acquisition phenomenon that has gathered truly remarkable momentum. Started by Jimmy Wales in 2001, Wikipedia defines a Wiki as the collaborative software and resultant web forum that allows users to add content to a website and in addition, to collaboratively edit it. The Internet, and indeed Wikipedia, lets millions of people collaborate in creating, linking (organising), and refining knowledge. Wikis and Wikipedia provide a radical extension to the blog, web forum, and folksomony paradigm.

[92] http://wikipedia.org/

Further to this, Völkel et al. (2006) provide a recent discussion of Wikipedia (Völkel et al., 2006 p2) in which modifications are suggested to enhance its semantic searchability.

At an invited talk at OOPSLA / WikiSym in Oct 2005, Wales referred to the Wikipedia phenomenon as the "free culture revolution" (Wales, 2005)[93]. At http://wikipedia.org, when anyone in the public Internet community (including you and me) feels that a particular topic requires content, they are able to insert the topic into the knowledge base, and generate the content. With more than 5 million articles in more than 200 different languages[94] it is a community project to create a global encyclopaedia spanning every topic in every language. Already it provides a phenomenal resource in areas of history, science, engineering, the arts, politics, and society.

The Wiki philosophy (Cunningham and Leuf, 2001) reflects the belief that most times, knowledge (i.e. true or false knowledge) is better out in public than kept in private. With Wiki, an unwritten assessment has been made that private knowledge is of much less value than public knowledge, even when that knowledge is false, since publicly amendable knowledge that is false is more likely to be corrected than false private knowledge. By providing a collaborative forum for knowledge acquisition, knowledge conflicts are resolved more rapidly than such conflicts might otherwise be, and useful knowledge acquisition is more likely to occur. This correlates with the views of Zondag and Lodder (2005, p5) expressed earlier (section 5.2, page 66).

The Encyclopaedia Britannica and Wikipedia story summarised in Appendix J (commencing on p 417) tells of a revolution in knowledge management bought about by the ubiquitous Internetworking of computers and a massive leap in faith about the desire and ability of humans to collaborate and communicate. It demands a fundamental change in our beliefs about the scope and possibilities of collaborative knowledge acquisition, storage, and re-use. The story highlights the present demand for knowledge to be findable, elastic, current and relevant. It highlights the importance of drawing on a number of experts to contribute the knowledge dynamically.

---

[93] The international symposium on Wiki's is described at http://www.wikisym.org/. Coverage of OOPSLA 2005 talk has been provided at: http://wiki.cs.uiuc.edu/OOPSLA05/Wikipedia+in+the+Free+Culture+Revolution

[94] http://en.wikipedia.org/wiki/Wikipedia

While there has been controversy over and criticisms of the wiki approach[95], an important observation is that the approach lets multiple experts learn from, contribute to and refine the knowledge presented. It lets users continuously contribute to the best of their knowledge, so that the best knowledge of all users is the one that persists.

Finally, in Wikipedia, not only is the knowledge itself changing, but also the anchortext referring to that knowledge, and referred by that knowledge.

5.6 The Semantic Web

In the last several years the effort of the Semantic Web community to develop approaches[96] that allow mature ontologies to be built, distributed and interchanged, has provided key developments in the area of knowledge representation, reasoning, and visualisation. A summary of some of the semantic web languages is provided in Appendix K (commencing on p 419). As described in Appendix K, the Web Ontology Language: OWL (which builds on the XML-based RDF language) allows content in the semantic web to be collaboratively indexed by a distributed community of Internet users. Hence advocates for the semantic web are currently attempting to augment the anchor-text phenomenon used by popular search engines and the anchor-text phenomenon promoted as tags in folksomonies to provide a more context rich index for Internet addressable knowledge.

On a different angle, Beydoun et. al (2007) offer a novel approach to evolving a semantic web using social navigation – as mentioned earlier, they capture virtual surfing trails from cooperating web surfers. They too provide a review of the semantic web and the collaborative ontology literature (Beydoun et. al, 2007, p2).

---

[95] http://en.wikipedia.org/wiki/Wikipedia#Criticism_and_controversy

[96] The main effort of current research in knowledge representation is providing theories and systems for expressing structured knowledge and for accessing and reasoning with it. *Description Logics (see* http://www.cse.unsw.edu.au/~cs4418/Lectures/dlnotes/description-logics.pdf) are currently considered by the International Knowledge Representation and Reasoning community (see http://www.kr.org/KR2006/) to be the most important knowledge representation formalism unifying and giving a logical basis to the traditions of Frame-based systems, Semantic Networks, KL-ONE-like languages, Object-Oriented representations, Semantic data models, and Type systems (http://dl.kr.org/, April 2006). For information about inference in description logics, and the Tableaux Algorithm see (Baader and Sattler, 2000) (Baader and Nutt, 2003).

5.7 Chapter Summary

The chapter began by recalling that troubleshooting in the ICT support centre domain requires wide spans of knowledge from multiple and sometimes conflicting sources of knowledge. In answer to the research question:

Q3. What literature and technologies are relevant to a collaborative MCRDR-based troubleshooting approach?

a review of literature relevant to collaboration in knowledge-based systems discussed the conflict resolution issues relevant to a multi-user approach.

The chapter went on to explore recent technology trends in "new age" web-based collaborative knowledge acquisition. Trends in the use of Anchor-text and Folksomonies together with current Semantic Web approaches indicate that it is possible to create a KBS that relies on some form of article tagging by multiple users to increase the findability of knowledge for both individuals, and groups. The review of Wikipedia indicates that it is possible to create a KBS that lets multiple users concurrently and incrementally contribute to the best of their knowledge, so that in general, the best knowledge of all users is the one that persists.

As discussed previously (section 4.2 on page 40), the philosophy underpinning RDR is that knowledge is socially situated, contextual, continually changing, and emergent (Compton and Jansen 1989). In brief, knowledge as something made up to fit the given context. Given that knowledge is socially situated, and that people themselves form part of the context in which the knowledge is to be captured and reused, it seems natural that a Collaborative MCRDR (C-MCRDR) approach, as proposed by this research, should emerge.

Based on the analysis of solution retrieval problems facing the support centre described in Chapter 2, and as described in Chapters 3 and 4, it appeared that the MCRDR paradigm could provide a useful platform on which to base a KBS for the support centre. However, the MCRDR structure and algorithm would need to be significantly extended to allow multiple experts to update the knowledge base, and to facilitate the acquisition of knowledge from possibly conflicting sources of expertise. The system would need to support collaborative case and problem definition; problem determination and classification; and solution definition.

Allowing multiple experts to collaborate and resolve their classification conflicts within an MCRDR framework could provide the benefits of a Wikipedia-style collaborative KA forum. The aim would be to create a KA forum where multiple experts can contribute to the best of their knowledge, so that the best of everybody's knowledge is the one that persists. As noted by (Beydoun et. al, 2005, p65) "*a state of convergence between experts ... is assumed to reflect the world more accurately*".

Hence a solution could be developed that is analogous to the Folksomony idea where multiple users tag cases with lexical symbols meaningful to both themselves and their colleagues. This would parallel the anchor-text phenomenon exploited by popular search engines. As shown in Table 7 on page 74, in a multi-expert MCRDR paradigm, the tags would be comprised of rule conditions that can be used to classify incoming problem cases and link them to relevant solutions. Separate "truths" i.e. separate viewpoints would need to be maintained in the KBS – the private viewpoints of individual contributors, as well as the combined overall and public viewpoint of the group.

The next chapter provides further review of conventional MCRDR systems in light of a multi-user problem-solving context.

CHAPTER 6: ADAPTING MCRDR

6.1 Chapter Outline

The PEIRS trials (Edwards, 1996), the early LabWizard trials (Edwards, 1996), and PKS's implementation of LabWizard at the time of this research (see section G.10 on page 411) targeted problem domains with the following characteristics[97,98]:

- The cases came from the outside world – they were the results of pathology lab tests taken on blood and urine samples. The KBS could not change these results, but the KBS could conclude that the combination of results looked peculiar and the lab should check them. In summary, the cases were unchanging.
- Only one expert was called upon to update the knowledge at a time.
- It was felt that few errors were made, possibly because the domain had been mapped several times previously, and the characteristics of the domain were well-known[99].
- Redundant knowledge wasn't a big a problem since cases were generally processed offline and in batch mode i.e. in a non real-time environment. On-demand one-off case-specific interpretations could be made as an exception, rather than as a rule.

---

[97] In the PEIRS trial reported by Edwards, cases in the form of laboratory test results were sometimes created with errors, and needed correction (Edwards 1996, p92). However, the patient records referred to were on a totally separate and previously established laboratory information system and could not therefore be modified. As well, new domain knowledge would sometimes come to hand, requiring that existing knowledge be corrected (Edwards 1996, p163).

Because PEIRS was experimental, it was allowed only very limited interaction with the LIS. Once a case or its associated knowledge had been corrected, the interpretive comment previously associated with the case would become invalid. So in PEIRS, immediately after printing interpreted cases, an *erase* program was automatically executed to remove comment codes from all of the cases that had been printed. This was inconvenient because the interpretations for all cases would be erased and they wouldn't be available online for others to see (Edwards 1996, p 122, 131).

[98] Many thanks to an anonymous PKAW 2006 reviewer of the (Vazey and Richards, 2006b) paper for some of these very helpful reflections on the research presented.

[99] In fact, even the pathology domain continues to evolve as new sub-domains and knowledge come to hand.

Similarly in the NRDR work (Beydoun and Hoffman, 1997) (described in section 4.4.5 on page 62), the case was the chess game so far – the KBS could not change the history of the game; only one expert was required to update the KBS, and processing was done offline.

Clearly these previous RDR systems dealt with different sorts of cases in a structurally different domain to that being dealt with by this research. The troubleshooting domain reported in Chapter 2 introduces new challenges for the MCRDR algorithm[100]:

- Cases may be constructed on the fly (by multiple different experts) as information about the nature of the problem comes to hand.
- Many experts are needed to create the knowledge (i.e. the conditions, classifications and conclusions) in order to maximise coverage of the domain, and the completeness of the knowledge base. No one expert holds all the requisite information.
- The uncertain nature of cases and RuleNodes means that the edit of Cases, RuleNodes, and their associations needs to be supported, both as the KBS is built and during routine operation.
- The KBS needs to be efficient (and hence compact) so that it can support the rapid real-time responsiveness required by users.

Hence this chapter begins to address the following research question:

Q4. How can MCRDR be adapted to a real-time environment where problems and their solutions are continuously evolving, and where a distributed group of stakeholders can both contribute to and benefit from the acquired knowledge?

The chapter provides an introduction to the ways in which this research has adapted the MCRDR algorithm initially developed by Kang (1995) to the trouble-shooting context in the support centre domain.

[100] Relatively recent work by (Wada et al., 2002) and (Yoshida et al., 2002, 2004) adapts SCRDR to a different novel problem domain. The work proposes the combining of KA from human experts with inductive learning from labelled data. A facility to delete RuleNodes is also suggested.

6.2 Support Centre Challenges for MCRDR

The support-centre environment has a number of properties that collectively present new challenges for the MCRDR algorithm[101]:

- The system needs to deal with numerous cases - in the order of 5000 per day globally;
- The volume of cases being dealt with means that the workflow must inherently deal daily with decentralised system maintenance and knowledge acquisition – hence multiple users will use and update the system, but a limited subset of privileged users may need to approve their updates;
- Initial problem descriptions are sparse – the case definition matures as customer service personnel interact with the customer and *work the case* - multiple users may need to describe the cases through an interactive question-answer interface to the system that will assign the relevant A-V pairs to the case;
- While most cases are resolved promptly, a number of cases are open for days or even weeks;
- Problem receipt and resolution is asynchronous since there is a time delay between when the system receives a problem case, and when a customer service representative can attend to it;
- The system must interact with a legacy ticketing system and legacy knowledge base;
- Archived cases and the conclusions registered to them need to be available for several years (perhaps 10 years for some cases) into the future;
- Old cases/ old conclusions may be edited;
- The granularity of conclusions may vary widely and conclusions that are web links may expire;
- Very many attributes will exist and vary across cases and new attributes will frequently need to be added;

[101] Some of this text appears in (Vazey and Richards, 2005a).

- The range of values possible for those attributes may be very large and the dependencies between these attribute-value (A-V) pairs may be very strong. For example, issues may span multiple systems, platforms, vendors, versions, etc.
- The proposed system will not have control over the referred case data, which is stored in the parent company's database;
- The attribute-value (A-V) pairs and rules in my system cannot be simple keywords, and simple tests for existence of keywords. Rather, the attributes may be any type e.g. integer, float, string, enumerated type, or free-text; they may be single valued, one of a set, or some of a set; and tests may include tests for range such as 'installation date > 2001/01/30'; for existence (indicated as ?) such as '? patch 3.6.5'; for containment e.g. 'case description contains *machine generated*' or for equivalence e.g. 'version == 3.2'.
- The system needs to fit smoothly into the workflow of a bustling call centre – expediency, efficiency and accuracy will be key to the system's success.

The remainder of this chapter proposes some novel modifications to the MCRDR approach in support of these domain characteristics.

6.3 Supporting Case Change

Users in the support centre domain have to define the cases, as well as the rules (described previously in section 4.3.2.3 on page 58). In a changing and uncertain domain, MCRDR cases, including cornerstone cases, may be constructed with numerous errors or omissions, particularly by novice users, but also by experienced users.

Cases may require change during knowledge building, as well as during routine use. In the support centre environment, the knowledge evolution – knowledge building cycle would be ongoing, even for previously chartered sub-domains[102].

At this point, the terms *live* and *registered* need to be introduced. In the proposed system, when a RuleNode is *live* it means that this RuleNode is the last `TRUE` RuleNode for this case

[102] Although in mature knowledge sub-domains, the KBS administrators may decide to restrict access to some RuleNodes via the user profile and RuleNode approval mechanism – this feature in the 7Cs system proposed later in the thesis (Chapters 9, 11, and 13) is discussed in Appendix O.1 on page 451.

in one of the paths through the rule tree and hence it represents one of the current classifications for the case on hand. In contrast, when a RuleNode is *registered* it means that at some time in the past a user acknowledged and hence *registered* that the RuleNode was *live* for the current case. This separation between live and registered RuleNodes was presented to HTG (Vazey, March 2004) as well as in (Vazey and Richards, 2004a), and it later formed the basis of the US patent application described on page 283.

In the system proposed by this research, the modification, addition and deletion of attribute data for cases already seen and evaluated by the knowledge base is supported. The same case is re-evaluated by the KBS, and by tracking the difference between RuleNodes previously *registered* to the case, and RuleNodes currently *live* for the case, a change history is maintained showing how the classifications and hence conclusions of the case have evolved over time.

6.4 Supporting Cornerstone Substitution

Conventional SCRDR systems only recorded one cornerstone case per RuleNode, being the case that caused that RuleNode to first be created[103]. As discussed previously in section 4.3.2 on page 54 and also in (Richards, 1998a, p55) and (Kang et al., 1995, p5, p10), in MCRDR systems the *set of Cornerstone cases* for a RuleNode is comprised of all the Cornerstone cases for that RuleNode and its dependent RuleNodes.

If a cornerstone case changes, then it may not be relevant to keep it as the Cornerstone case for that node. Since cases could not be changed, conventional MCRDR systems did not provide a mechanism for substituting cornerstone cases in this situation, or for tracking the changes to existing cases, and monitoring the effect of those changes. Further to this Gaines (1993) suggested that all cases could be kept in the system to improve KA.

In the system proposed by this research, after knowledge evolution occurs, if there is another case that would now make a better and more representative Cornerstone case for a given RuleNode, then that case is substituted as the new Cornerstone case at that RuleNode. (Note that at the time of substitution, this case need not be a cornerstone case for any RuleNode in

[103] In this thesis, the conventional SCRDR definition of the term "cornerstone case" is used. Please refer to the glossary on page 277 for further clarification.

the system – it's just a case that is currently *live* (and possibly also *registered*) for that RuleNode.)

6.5 Tracking Case drop-throughs

As a KBS is updated with new knowledge (including changes to RuleNodes, rule conditions, classifications and / or conclusions), the KBS can change in such a way that a case previously evaluated by that KBS will now arrive at different conclusions. The term "case drop-through" refers to the specific scenario where a case that used to be *live* for a parent RuleNode, stops being *live* for that node, and becomes *live* for the child RuleNode instead. For those unfamiliar with the case drop-through terminology, an example of the problems of case drop-through is provided in Appendix P (page 468).

Edwards (1996, p88) highlighted the case drop-through problem in conventional SCRDR and noted the potential to corrupt the KBS. Similarly, Kang (1995, p50) identified the situation where knowledge needs to be changed in such a fashion that a cornerstone case for an existing RuleNode will drop-through to a new child RuleNode. Kang also suggested that if absolutely necessary, the rules suggested by the MCRDR difference list for the new RuleNode could be overridden (Kang, 1995, p65). However the strategy for deciding the replacement cornerstone case for the parent RuleNode once its cornerstone case has dropped-through, and for determining the cornerstone case at the new child RuleNode was not fully explored (Kang, 1995, p67).

Chklovski (2001, p15) highlights that successful knowledge acquisition has to be transparent rather than opaque. He argues that it should be possible to understand why a system did what it did, to see the impact of contributions, and to correct the behaviour of the system. Similarly Beydoun et. al (2005, p64) argue that "*internal inconsistencies, or clashes between components of a model, are a reflection of its incompleteness*".

Hence in the system proposed by this research, a robust and transparent strategy is provided to handle the case drop-through scenario (described later in section 11.7 on page 210). By maintaining a separate record of the *live* and *registered*[104] case-RuleNode associations, a difference list can be formed that highlights situations where case drop-through has occurred.

[104] If required, please refer to the glossary on page 277.

This strategy supports verification and validation[105] of the KBS. For example, when case drop-through occurs case-Rulenode associations that were previously *live* and *registered* will suddenly no longer be *live*, but they will still be still *registered*. This difference between the live and registered case-RuleNode associations provides an additional and previously untracked knowledge acquisition opportunity for users so that KA can occur much more rapidly (for some examples, see Figure 109 on page 433 and Figure 110 on page 434). Later on when the knowledge base is more mature, cases can be untracked (described in Appendix O.4 on page 454) so that unnecessary notifications are eliminated.

Kang, Gambetta and Compton (1996, p264) showed that for the GARVAN-ES1 data, the *machine-detectable error* introduced to cases already seen by a single-expert SCRDR system was 2% at the commencement of KA. It declined thereafter to the same size as the final error on unseen cases (less than 0.5%). The machine-detectable errors resulted from misclassifications and was caused by over-generalisations, under-generalisations, or omissions in the immature (but evolving) knowledge base. Similar results were reported in (Compton, 2000). However, the *human-detectable-error* might be much greater than the *machine-detectable error*. The *human-detectable-error* may include additional errors such as over- and under- generalisations not detected by machine interpretation of the cases seen thus far, poor quality or inconsistent classifications or interpretations, references to irrelevant attributes, unresolved conflicts between differing expert opinion, and partial or complete replications in the knowledge base (Gaines, 1989, p1). Tracking classifications for past cases can highlight even more of these *human-detectable-error*s by highlighting previously undetected situations where case drop-through occurs.

In a multi-expert system, the benefit of tracking classifications for past cases may be even greater than for a single-expert system since the differences in expert opinion will be discovered and communicated much more rapidly. Multiple experts will allow more sub-domains to be covered, more broadly, and the benefit will be repeated for each new sub-domain added to the KBS.

[105] For a discussion of verification and validation with Ripple Down Rules, see (Kang, Gambetta, and Compton, 1996).

6.6 Direct Edit Facility

In previously tackled domains such as pathology, it was felt that experts populating RDR KBSs were unlikely to make repeated errors. For example:

> "*It should be noted that we are supposedly dealing with experts so that it makes little sense to speculate about the possibility of repeated gross errors of expertise.*" (Kang, Gambetta, Compton, 1996, p261).

Frequent (human and / or machine detectable) error possibly was not a problem with the SCRDR or MCRDR implementations of PEIRS since the system was only ever maintained by a single user (Glenn Edwards), and that user already had significant experience in codifying a large part of the chosen problem domain (Thyroid diseases). Edwards was codifying the thyroid domain back in 1984 with GARVAN-ES1 so that domain provided mature codification territory during his later RDR trials (Edwards, 1995 p 15).

In contrast, in the support centre domain where the domain knowledge is uncertain and changing, this research found that repeated gross errors were frequent. In the software trial of the 7Cs system proposed by this research, out of 172 case creations, a further 139 case edits were required, and out of 107 RuleNode creations a further 141 RuleNode edits were required (Table 21 on page 224).

The frequency of error is obviously dependent on the nature of the problem domain, the data collected for that domain, and the knowledge and experience of the experts codifying the domain. For population of a knowledge base by a group of human users who make spelling mistakes as well as logical and classification errors in a previously unchartered problem domain[106], the number of errors and subsequent case drop-throughs over time could be very significant. For example, Easterbrook (1991, p2) highlights that conflicts can and do arise from a thin spread of application domain knowledge; fluctuating and conflicting requirements; and breakdowns in communication and co-ordination. He refers to these scenarios in the context of software engineering, but it is clear that they apply to knowledge elicitation in many different domains. Similarly, Horn (1993) refers to the work of Lientz and Swanson (1980) suggesting three major maintenance categories for expert systems: corrective

[106] or alternatively by a group of users without any expertise as simulated by (Compton, P., Preston, P. and Kang, 1995)

maintenance that deal with “bug fixes”; adaptive maintenance that deals with our changing world; and perfective maintenance that deals with changing user requirements. Van Vliet (1999) adds preventative maintenance.

The SCRDR evaluation by Compton, Preston and Kang (1995) referred to by Compton (2000) used machine-learning techniques to construct knowledge bases using data from the Irvine Data Repository (Chess, Tictactoe, and Garvan) and then used those learnt knowledge bases as the gold master in acquiring knowledge. However, the Irvine data sets have incomplete case data, and possibly classification errors as well. For example see the comment "Plenty of missing data" for the "Thyroid Disease Database" at http://www.ics.uci.edu/~mlearn/MLSummary.html. Without human expert input it is really impossible to estimate the volume of errors and subsequent case drop-throughs that might occur if the resultant knowledge bases were corrected.

In the support centre domain, since this was the first time that the knowledge was being acquired, it seems reasonable to expect that errors would occur. The most effective and efficient strategy for managing these errors wasn’t necessarily to patch (in the case of over-generalisation errors) or duplicate (in the case of over-specialisation errors) that segment of the rule tree. User trials indicated that users want and need to be able to edit all aspects of the KBS, including cases, conditions, classifications and conclusions (see section 12.10 on page 237). As well, users need the KBS to be able to highlight inconsistencies that occur in the case drop-through scenario, and inform affected participants in response, so as to maintain and promote integrity over the knowledge. For related work on reasoning in the presence of conflicts and inconsistencies, see the work of Cheung on paraconsistent reasoning as reported by Bauer (2005).

6.7 Supporting Case Construction / Configuration

In the support centre domain, this research found (section 2.5.3 on page 26), that the expert needs to first “work-up” or build a case through an iterative process of fetching and recording more and more case detail. As discussed previously (section 4.4.4in page 61), I-RDR was intended to be a query mechanism. It was not concerned with the acquisition or maintenance of cases or rules. As well, R-RDR was intended as a computer-driven configuration mechanism (section 4.4.2, page 61). But the ability for a human user to “work-up” a case that changes over time and becomes part of the system was not supported in conventional

MCRDR systems. The system proposed in this research significantly aids this process by supporting case configuration using an interactive and recursive multiple classification approach.

Edwards (1996, p18, 38) refers to Szolovitz's suggestion that MYCIN[107], one of the earliest medical expert systems, demonstrated that complex flowcharts could be reproduced by a vastly smaller handful of rules and a simple recursive algorithm. Hence the ability to work-up a case can be seen as the ability to recursively answer questions in a complex workflow, and re-evaluate the case to discover new paths through the flowchart.

In the proposed system, interactive and recursive MCRDR functionality (IR-MCRDR) is provided so that one or more of the evaluated classifications for a case may have one or more conclusions that are requests for the user to more precisely specify the case by entering additional case data, or directives to the KBS to construct (i.e. set) additional case data. Once this data has been provided the case will be re-evaluated against the rule tree. On re-evaluation of that modified case it is possible that another path will be found through the rule tree and that the case will fetch a new set of conclusions. Some of those new conclusions may again be requests for additional case data. Boose et al. (1992, p2-4) described the process of building up a solution from component pieces as *synthesis* i.e. generative or constructive; as opposed to just *analysis* which they say involves identifying sets of objects based on their features. Hence, this may appear to the user as a process of guided case construction and guided classification or problem determination through the multiple classification KBS.

6.8 No Notification of changes to the Knowledge.

As discussed previously in section 6.5 (page 84) , in a dynamic knowledge environment where the knowledge-base can move and change, over time a case may satisfy different rule conditions, and hence it will fetch different classifications and conclusions. In conventional MCRDR systems, the notion of registering the case-RuleNode associations did not exist and thus the live versus registered status of the associations was never recorded or tracked.

[107] MYCIN was an expert system developed for diagnosing acute meningitis and recommending appropriate antibiotic therapy.

As well, conventional MCRDR systems didn't allow cases to be "Tracked" in the sense that they don't automatically re-evaluate the conclusions for cases when the knowledge base changes. Conventional MCRDR systems allowed a case to be evaluated against the knowledge base, and the conclusions determined. But then the case was "let-go". There is no automated process by which a party interested in that case can be notified when the knowledge in the rule tree moves on in such a fashion that the case is now fetching a new set of conclusions. The value of keeping a history related to cases is further demonstrated in the example provided in Appendix P on page 468.

In the proposed system, users can be automatically notified whenever knowledge of interest to them changes.

As discussed in section 6.5 (page 84), when case drop-throughs occur case-Rulenode associations that were previously *live* and *registered*[108] may no longer be *live*, but they will still be *registered*. This difference between the live and registered case-RuleNode associations provides an additional and previously untracked knowledge acquisition opportunity for users so that KA can occur much more rapidly. Users could register their interest in particular RuleNodes or cases in the system, for example all items previously created and/or edited by that user, or all items dependent on a particular RuleNode in the KBS. Then when a relevant case-RuleNode association changes, the user can be automatically notified that a new KA opportunity has arisen e.g. via email or SMS. The consequent KA could involve registering new case-RuleNode associations or unregistering old case-RuleNode associations for a case, modifying the conditions under which the affected RuleNodes will fire, changing the classification labels or conclusions at the relevant RuleNodes, or modifying the affected cases.

6.9 Lost links between General and more Specific Classifications

In conventional MCRDR, if a user wanted the KBS to present the conclusions of a more general RuleNode, together with the conclusions of a more specific RuleNode for a given case, then two separate and possibly somewhat duplicated paths would be required in the

[108] If required, please refer to the glossary on page 277.

decision tree in order to allow the two different classifications to both fire for that case[109]. This approach has two shortcomings: the relationship between the more general superclass and the more specific subclass is lost, and in many cases, redundant knowledge will be stored. While redundant knowledge may not have been a significant problem in the pathology domain where processing of cases was done in batch mode, in the support centre problem domain, the KBS needs to be compact and efficient so that it can support the rapid real-time responsiveness required by users.

Hence, in the proposed system, users can assign labels to classifications and users can create links from child nodes to more general parent nodes using these labels[110]. In that way, the conclusions of more general parent and other nodes can be referenced and/or reproduced at RuleNodes that additionally offer more specific conclusions. In the proposed system, this is done by a refer() function as described later in section 11.3.6 on page 193.

6.10 Chapter Summary

In this chapter, enhancements to the MCRDR approach were suggested in order to begin answering the following research question:

Q4. How can MCRDR be adapted to a real-time environment where problems and their solutions are continuously evolving, and where a distributed group of stakeholders can both contribute to and benefit from the acquired knowledge?

The need to support users in their collaborative configuration (construction) and update of Cases and RuleNodes (including the conditions, classifications and conclusions) in the support centre problem domain was presented.

---

[109] For example, if a rule node concludes that the case represents problem class A and the expert wishes to say the case more specifically represents problem class A.1, then the user must add a new rule higher up the tree with possibly several duplicate rule conditions to conclude that the case is also type A.1. This is because in conventional MCRDR the conclusions on parent nodes are always over-ridden by `TRUE` child nodes.

[110] Although N-RDR (previously discussed in section 4.4.5 on page 62) (Beydoun and Hoffman, 1997) allowed rule conditions to be use the intermediate classifications provided in separate knowledge bases, the mechanism did not allow classification labels to be referenced and displayed in the conclusions at RuleNodes elsewhere in the decision tree.

As well, the concepts of *live* versus *registered* RuleNodes were introduced, together with the concept of *tracking* these case-RuleNode associations as the knowledge evolves. As the knowledge base evolves, the benefits of separating the *live* versus *registered* views of the knowledge include the ability to:

1. Notify collaborating users of changes to the knowledge base that might affect them,
2. Support the users with a more responsive real-time KBS by using a background processing mode to keep the knowledge up to date, and
3. Maintain relevant cornerstone cases, for example when case drop-through occurs.

Finally, the refer() function was introduced so that more specific RuleNodes need not duplicate information provided at more general RuleNodes, and so that separate paths are not required through the decision tree to arrive at the separate but related classifications.

Later on and in further response to the above research question, the top-level design, detailed design, implementation and software trial of the proposed 7Cs system are presented. However, before looking further at how this research question has been addressed, the next chapter takes a look at case-driven and rule-driven KA in general. A mathematical model is derived showing the expected probabilistic trajectory of knowledge acquisition in a case-driven knowledge acquisition (KA) paradigm. This is compared with the expected trajectory in rule-driven KA paradigm in order to provide some fundamental insights into case-driven KA, and emphasise the benefits of a hybrid case-driven and rule-driven KA approach.

CHAPTER 7: A MODEL OF KNOWLEDGE TRANSFER

7.1 Chapter Outline

This chapter addresses the following research question:

Q5. What is the expected trajectory of the case-driven acquisition of classification knowledge?

As discussed previously (section 3.5.1, page 36), conventional rule-based expert systems give primacy to the classification, rather than the context (Edwards, 1996, pp 170 - 171) whereas case-based KA systems like RDR give primacy to the context, rather than the classification (section 4.2, page 40). Further to this, bottom-up case-driven KA is compared with top-down rule-driven KA, and a stochastic model for case-driven KA is derived. The derived model provides a formula that allows predictions to be made about the rate of case-driven KA. The derived formulas have recently been published in (Vazey, 2006a) and some of the derivations have been published in (Vazey, 2006b).

The mathematical model supports the view that for optimal knowledge transfer, a hybrid case-driven and rule-driven KA approach is required. The value and importance of a hybrid case-driven and rule-driven approach to KA is explored further in Chapter 8 (page 133).

The derived model is relevant to Case-Driven Knowledge Acquisition (CDKA) for example in Artificial Intelligence, Machine Learning, Data Mining, Expert Systems, Ripple Down Rules, Group Decision Support Systems, Collaborative Tagging, Folksomonies, and Case-Based Reasoning (CBR) systems. The case-driven KA trajectory[111] reflects the natural slowing of knowledge exchange in an environment where a finite set of classes is mapped to the incoming search criteria or semi-structured cases by keywords or rule conditions as in Table 7 (page 74). The case-driven KA model offers important predictions for the trajectories presented in previous machine-learnt and case-based KA simulations for SCRDR and MCRDR as discussed in (Compton, Preston and Kang, 1995), (Kang, Lee, Kim, Preston and Compton, 1998), and (Cao and Compton, 2005 and 2006) as well as the tag-acquisition

[111] In this context, the term "trajectory" means the graphical depiction of the number of classes acquired in relation to the number of cases seen by the KA system.

trajectories discovered by (Goldman and Huberman, 2005, p4) for folksomonies. A separate and loosely related approach at modelling the maturity of incremental cooperative (as opposed to collaborative) KA was provided by (Beydoun et. al, 2005). Another loosely related approach known as Rated MCRDR (section 4.4.7 on page 63) uses a neural network back-end to identify inadequacies in a knowledge base for a given case (Dazeley and Kang, 2004, Figure 4, p10).

As noted by Beydoun, Kwok and Hoffman (2000, p3): "*empirical RDR research dwarfs formal and theoretical analysis of the methodology*". In the future, it is hoped that the case-driven KA model presented in this chapter will allow the performance of different KA approaches to be fairly compared, and the effect of purely random case-driven KA data recognised.

7.2 An Analysis of Case-driven Knowledge Acquisition

As mentioned earlier (section 3.5, page 35), KA is the process of acquiring knowledge from one or more third parties, be it from some individual or a machine, or from a group of individuals or machines. In this section we look at different types of knowledge.

One type of knowledge that can be acquired is class knowledge, including the conditions under which the class should be applied, the classification label for the class, and any conclusions associated with that classification[112]. Knowledge in the form of classes can be acquired directly as top-down generalised classes, or as bottom-up experience-based classes derived by examining specific cases. In this example, the acquired classes could apply matching keywords or rule conditions as in Table 7 (page 74) to map the incoming search criteria or semi-structured cases to their resultant classifications. The acquired classes that include the (keyword or rule condition) tags form an index between cases and their corresponding classifications.

7.2.1 SCRDR and MCRDR

In the evaluation of SCRDR by (Compton, P., Preston, P. and Kang, 1995), the case-driven acquisition of knowledge by multiple experts was simulated using 3 domains; 3 machine

[112] Other knowledge discussed later in the conclusions to the thesis includes the knowledge of relationships between classes.

learning algorithms; 3 simulated levels of expertise; and 6 randomizations, resulting in more than 180 result sets. Subsets of the Chess, TicTacToe, and Garvan Thyroid datasets from the Irvine data repository[113] were used. Similar experiments were performed in the evaluation of MCRDR by (Kang, Lee, Kim, Preston and Compton, 1998). Related and more recent simulations have been performed by (Cao and Compton, 2005 and 2006). When normalized, each of the result sets display similarly shaped trajectories for the number of RuleNodes (i.e. classes) acquired, and for the number of errors remaining in the KBS compared to the number of cases seen. Some of the results from Kang et al. (1998) are reproduced in Figure 8.

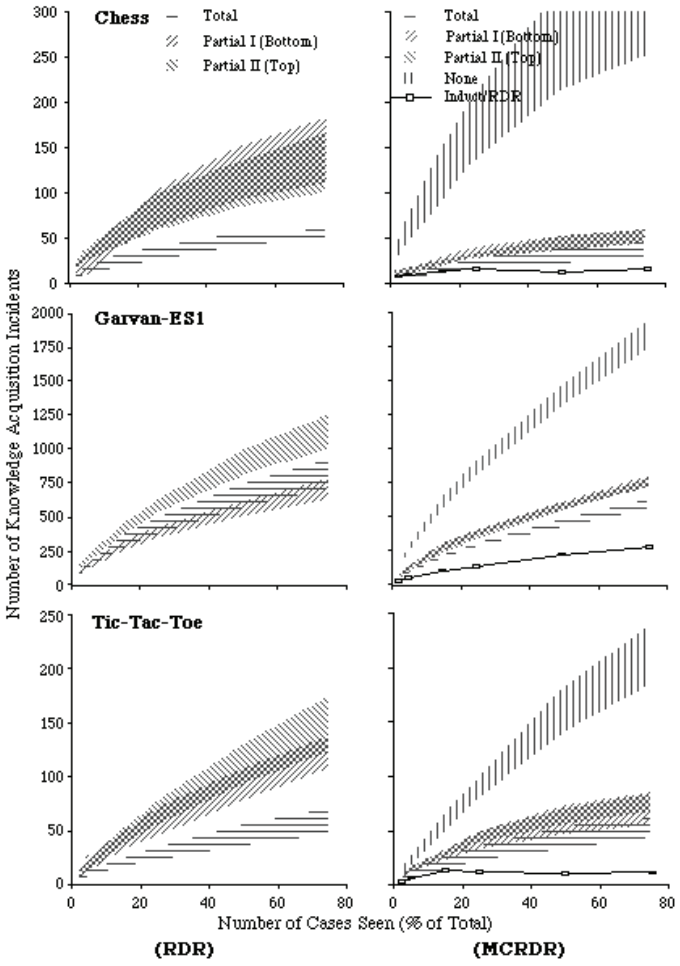
In Figure 8, the left-hand column demonstrates the trajectory of a machine-learnt SCRDR system across the three different knowledge domains, whilst the right-hand column of the figure demonstrates the trajectory of an MCRDR system across the same domains[114].

Referring to Figure 8, the x-axis shows the number of cases seen as a percentage of the total number of cases available, and the y-axis shows the number of RuleNodes acquired. Each of the result sets display trajectories that correspond with the expected case-driven KA and error trajectories derived herein.

[113] http://www.ics.uci.edu/~mlearn/MLRepository.html

[114] From Kang et al. (1998), but not critical to this discussion, the first test set ('Total') uses all the conditions in the Induct rule trace. The second test set ('Partial') uses only one condition from the intersection of the Induct rule trace and the difference list. As a minor refinement, since the order in which Induct selects conditions may effect the outcome, two extremes of selecting the single condition from the top ('Partial II') or bottom ('Partial I') of the rule trace were used.

Figure 8: SCRDR and MCRDR rules acquired versus cases seen



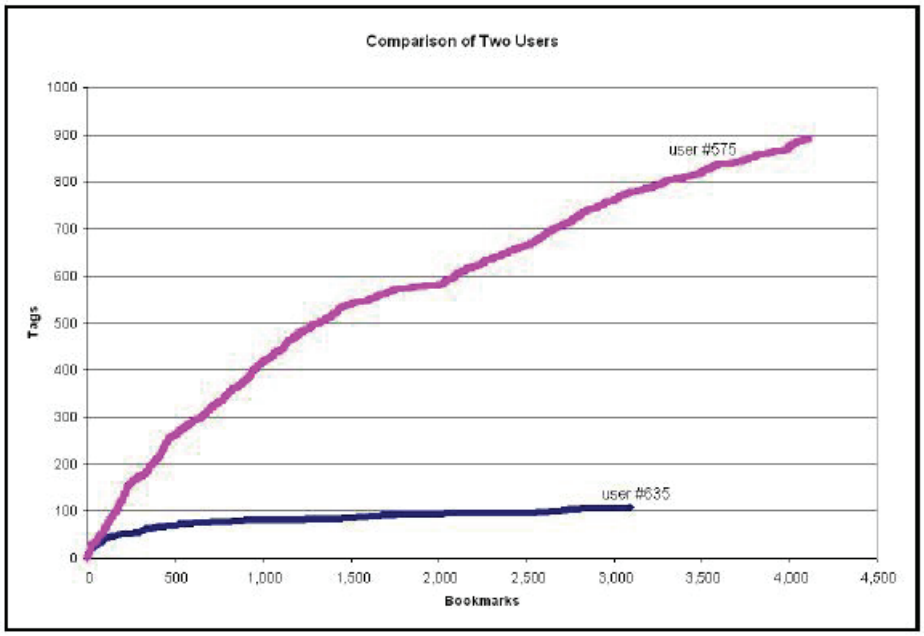
Reproduced from Kang, Lee, Kim, Preston and Compton (1998) with the kind permission of Byeong Kang.

7.2.2 Collaborative Tagging

As described previously (section 5.4, page 72), Collaborative Tagging, also known as Folksomony, describes the process by which many users add metadata in the form of keywords (i.e. anchor-text) to shared content (Golder and Huberman, 2005).

In Figure 9, two users with very different tag-usage patterns are tagging novel websites (i.e. bookmarks) on a website-by-website (i.e. case-by-case) basis (x-axis). As they bookmark more websites (x-axis), the number of novel tags (i.e. novel classes) that they use (y-axis) increases in the monotonically increasing but slowing case-driven KA manner.

Figure 9: Two extreme users' (#575, #635) tag growth. As they add more bookmarks, the number of tags they use increases, but at very different rates.



Reproduced from Golder and Huberman (2005, p4)
With the kind permission of Scott Golder and Bernado Huberman.

Obviously user #575 is much more specific with the tags that he or she applies to their bookmarks than user #635. As noted previously (section 5.2 on page 69), the repetitive versus novel nature of cases depends on the level of specificity demanded by users (Golder and Huberman, 2005, section 2.1)[115]. When the domain is first being constructed, the demand for differentiation may be weak, but as the domain matures, the demand for differentiation may become stronger as experts start to require greater specificity. Since learning can involve a process of moving from the general to the specific, and then back to the general[116], the total possible amount of classes that can be acquired by the KBS, and hence the repetitive nature of cases may even be seen to change over time, depending on the users' changing and individual needs. As noted by Compton, Preston and Kang (1995, p1), "*clearly any study using experts*

[115] For instance one user might classify a corgi and a terrier both as dogs, and another user may classify them separately as a corgi and a terrier.

[116] Note that from an induction point of view, learning is a process of moving from the specific (instance) to the general (concept).

needs to take into account the variability between experts as well as the difficulty of repeat experiments on the same experts, whereby they become more expert at contributing to a Knowledge Based System (KBS)".

With these human factors in mind, this chapter hypothesises that the rate of usage of tags (and hence classes) in SCRDR, MCRDR, folksomonies and tagsomonies corroborates with the case-driven KA trajectory predicted herein. Later (section 7.2.5, page 112) we will see that when more than one class is acquired per case (as in MCRDR systems), the x-axis shrinks in linear proportion as described in (Vazey, 2006) so that even more rapid acquisition of the classification knowledge occurs.

7.2.3 The Nature of Knowledge

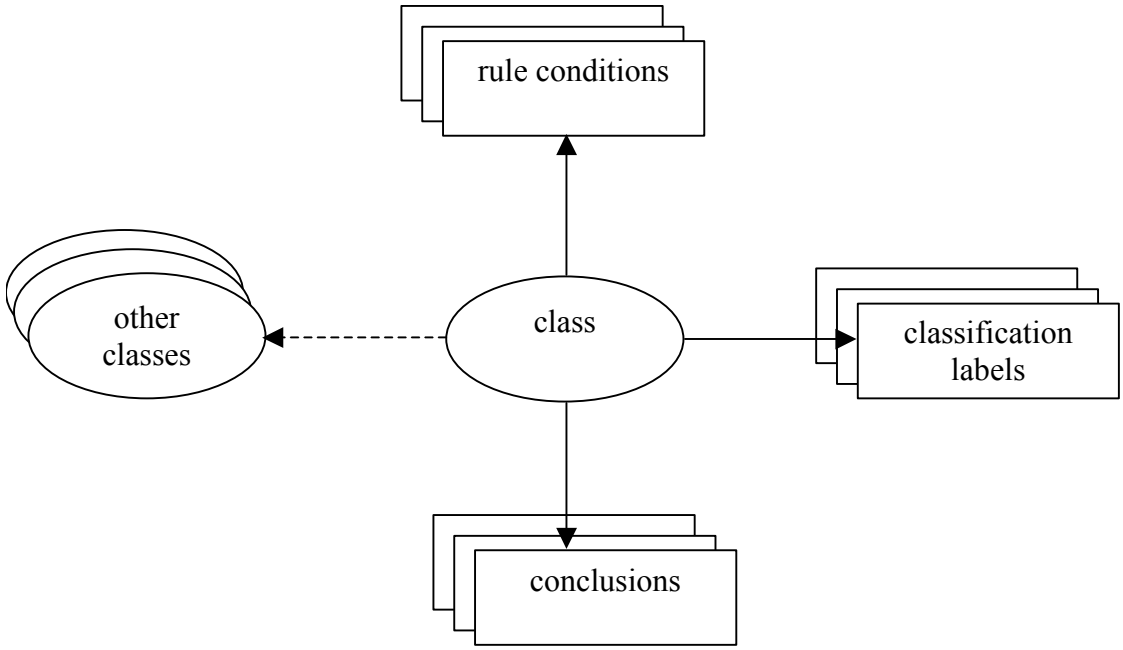
Knowledge transfer involves at least two parties and includes the codification, transmission, reception and consequent storage of the knowledge in question. In a case-based knowledge acquisition system, if some of the incoming cases are repetitive exemplars for identical classes, these cases will not present a new KA opportunity for the KBS and they will appear as repeats. The concept of the "number of pieces of knowledge" or *nuggets of knowledge* required to cover a domain is identified in Compton (2000, p7, paragraph 3). As well, (Cao and Compton, 2006, p2) highlight that the minimum number of rules required to model a domain is given by the number of non-overlapping i.e. mutually exclusive classifications in that domain. In this research, a *nugget of knowledge* is referred to as a *class*.

Typically a class will include the conditions under which cases will be true for that class, optionally the class or classification name(s), and the corresponding conclusions for that classification[117]. A class may also optionally include references to other classes. This is illustrated in Figure 10 on page 98.

[117] For conventional classification problems, in the past there hasn't been a separation between the classification and its related conclusions, and it has generally been accepted that the class label is the conclusion. However, in this thesis a different approach is taken since in practice, a classification may fetch multiple different conclusions, and those conclusions may be reused in combination or separately by other classifications in the same system. For example, consider medical practice where several different diagnoses share a common treatment, and where some of these diagnoses may simultaneously demand other treatments as well.

As indicated in the previous section the number of classes depends on how abstract or specific one is in describing the domain.

Figure 10: Relationship between Classes, Classifications, Conditions and Conclusions



In the proposed 7Cs system (described in more detail in Chapter 9, page 161), the *class* is represented by a RuleNode[118] located in a condition mesh that is comprised of the following: one or more representative Cases for that RuleNode (and not the dependent RuleNodes), the Rule Conditions (i.e. the tag conditions) that determine the Boolean outcome of the RuleNode, the Classifications[119] represented by that RuleNode, and the Conclusions for those Classifications. Rather than using a tree structure, in the proposed system a condition mesh is used that is analogous to a neural network (as discussed in Appendix D.7 on page 401, and

[118] There are as many classes in the system as there are RuleNodes. This is because each class includes not only the classification(s) represented by that class, but the means of identification for that class i.e. the associated rule conditions. Where a classification is repeated multiple times but under different rule conditions in the condition mesh e.g. in RuleNodes A_1, A_2, and A_3, these separate RuleNodes can be combined via a shared child RuleNode e.g. A to form a superclass that represents the conjunction of rule conditions of the subclass RuleNodes. This is explained further in section 13.4 (page 245).

[119] The classifications referred to at a RuleNode might include the target classification as well as any (inherited) ancestor classifications.

section 13.4 on page 247)[120]. Hence there may be multiple paths through the KA system to the same RuleNode. In the proposed system, the location of each RuleNode is therefore specified relative to multiple parent and multiple child RuleNodes.

In a folksomony the *class* represents the keyword-based tag that can be used to locate a relevant Internet resource. This analogy was previously discussed in Table 7 (page 74).

In SCRDR and MCRDR systems, the *class* is a RuleNode located in a decision tree that is comprised of a rule, a conclusion, and a representative case for that RuleNode known as the *cornerstone* case. As mentioned previously, in MCRDR a cornerstone case list for a RuleNode is derived using the cornerstone cases of all of its dependent RuleNodes. Since there is only one path to each RuleNode in these KA systems, the location of each RuleNode is specified relative to its sole parent RuleNode and its sibling RuleNodes (Chapter 4, page 40).

In human learning the *class* may represent one or more neurons or perceptrons and their location relative to other neurons in the same network. For a deeper discussion of the inter-relationship between neurons in the cerebral cortex of humans, and the proposed Recommendation Architecture (in which missing attribute values become implied) see Coward (2005). Alternatively, a nugget of knowledge might also represent the characteristics and location of abstractions in some high-level taxonomy, ontology or schema. For a description of schemas in human learning see Cooper (1998, section 3)[121].

[120] The 7Cs structure is analogous to a neural network since there are multiple possible inputs per case in the form of attribute-value pairs; multiple outputs in the form of classifications; and a mesh of RuleNode layers hidden from the read-only view of most users that permits each node to have multiple parent and child nodes with (in general) an acyclic evaluative flow through RuleNodes in the decision mesh; where the RuleNodes provide evaluative weights according to their rule conditions.

[121] Schemas are relevant to this research because they include high-level (and hence default) classifications and their conclusions, as well as low-level specific classifications and their conclusions. Note that figuratively speaking, in separating the wolves from the sheep in a schema or for example in an RDR, CBR, Data-Mining, or AI applications, it is just as important to know what a wolf is, as it is to know what a wolf isn't. Hence schema's can be regular decision trees that include exception conditions.

7.2.4 A Single Classification Case-driven Equal Frequency KA Example

Let's examine a case-driven knowledge acquisition scenario where cases are being mapped to just one class at a time (for example, as in an SCRDR system). Note that we begin by assuming that all the knowledge being transferred is true. (At a later stage, the reader can refer to Appendix Q on page 473 to see how the formulas change when some of the knowledge is false, speculative or unknown.)

Say that the target knowledge domain will be comprised of m classes that correctly map the incoming cases to their representative classifications and hence conclusions. In this study, N = 1000 cases were randomly generated[122], each comprising just one of m = 100 different classes. The m different classes were represented with equal frequency. For single classification case-driven KA as in SCRDR systems, each novel class represents a new RuleNode in the decision tree.

The number of times a case with a novel class was seen was cumulatively counted and the number of novel classes seen versus the number of cases seen was plotted[123]. Table 8 (page 101) illustrates the example and Figure 11 (page 102) shows the Actual trajectories for 5 independent case-driven KA scenarios, together with the Expected trajectory, and the Best Case straight-line trajectory for m = 100.

[122] In practice, and as discussed later on, users may self-select their cases in such a fashion that the order of presentation of classes to the KBS is not entirely random. In that case, KA may occur more rapidly than predicted by the single classification case-driven KA trajectory derived here. Many thanks to an anonymous reviewer of the Vazey(2006) paper for these comments.

[123] In previous research such as (Compton, P., Preston, P. and Kang, 1995), (Kang, Lee, Kim, Preston and Compton, 1998), (Compton, 2000), (Dazeley and Kang, 2004) and (Cao and Compton, 2005 and 2006), the process of matching cases to classifications been referred to as "simulating expertise".

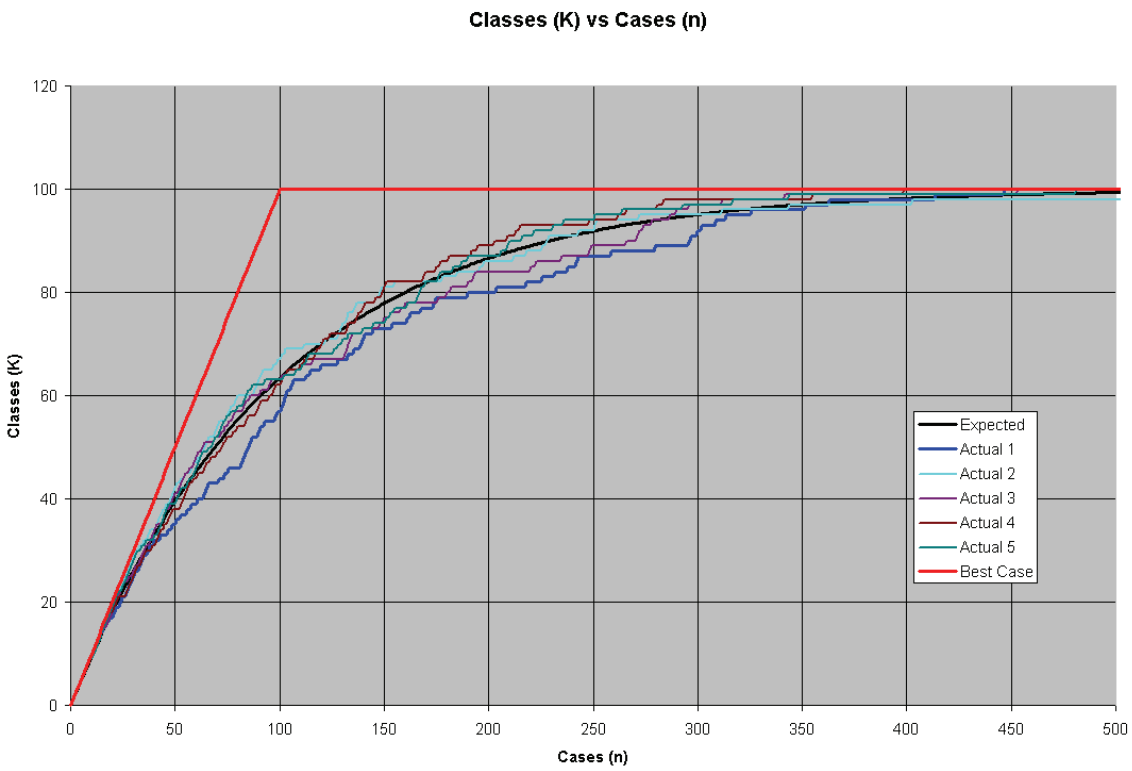
Table 8: Cumulative count of novel classes acquired in a Single Classification KA study

(Note: Data for the first 12 cases is displayed for a 100-class KA domain)

Case ID (1-1000)	Random class ID (1 – 100)	Number of times this class has been seen so far.	At this point, is the class novel? (1 = yes, 0 = no)	Cumulative count of novel classes acquired (<=100)
1	49	1	1	1
2	22	1	1	2
3	14	1	1	3
4	18	1	1	4
5	20	1	1	5
6	84	1	1	6
7	49	2	0	6
8	83	1	1	7
9	4	1	1	8
10	31	1	1	9
11	20	2	0	9
12	6	1	1	10
…	…	…	…	…

Every time newly generated random data is used a slightly different KA trajectory results with varying quantum KA steps. For instance on one draw, the system might fetch 50 consecutive cases all being exemplars for the same class, whereas on another draw the system might fetch 50 consecutive cases all being exemplars for novel classes. The varying trajectories represent the stochastic nature of the randomised incoming case data and correspond with the case-driven SCRDR KA trajectories shown in Figure 8 (page 95).

Figure 11: Expected, Actual and Best-Case Case-Driven KA trajectories.



In the next section a derivation is provided of the formula for the probabilistically expected case-driven KA trajectory for a randomised incoming stream of cases that is being mapped to a finite set of known and equally frequent classes. This trajectory is the smooth monotonically increasing and rapidly slowing asymptotic curved line shown in Figure 11.

(In contrast, in section 7.2.6 on page 118 the trajectory that one can expect to see when acquiring classes that occur with unequal frequency is derived.)

7.2.4.1 Classes Acquired as function of Cases Seen

The aim of this section is to prove the following theorem:

Theorem 1: The expected number of classes K_n that will be acquired after n cases by a single classification case-driven KA system is given by:

$$K_n = n - (1/m * \Sigma_{(i = 1 \text{ to } n)} K_{i-1})$$

where:

n is the number of cases seen;

K_n is the number of classes accumulated after n cases and $K_0 = 0$ and $K_1 = 1$; and

m is the total number of classes in the domain.

This theorem was used to construct the expected equal frequency single classification KA trajectory shown in the Figure 11 on page 102.

Proof 1: Proof of Theorem 1

Let the expected total number of classes acquired after the first case be K_1 and the incremental amount of knowledge acquired between the zero and first case be ΔK_1. For an initially empty knowledge base (i.e. $K_0 = 0$), and for case-driven KA the first case drawn will always be an exemplar for a novel class, hence the amount of knowledge acquired after one case will be:

$$K_1 = \Delta K_1 = m/m = 1$$

where:

m is the total number of classes in the domain.

The probability of the second case being novel depends on whether or not it is an exemplar for a previously seen class.

Since there are *m* classes to be acquired, and K_1 classes have already been acquired, the second case will be an exemplar for a novel class with a probability of $(m-K_1)/m$. Alternatively, the second case will be an exemplar for a class that is already known with a probability of K_1/m.

Using the weighted sum of probabilities the expected amount of classes acquired on the second case will therefore be:

$$\Delta K_2 = (m-K_1)/m * 1 + K_1/m * 0 = (m-K_1)/m$$

$$\text{where } K_2 = \Delta K_2 + K_1$$

Continuing along these lines we get:

$$\Delta K_3 = (m-K_2)/m * 1 + K_2/m * 0 = (m-K_2)/m$$

$$where\ K_3 = \Delta K_3 + K_2$$

By induction for case i we get:

$$\Delta K_i = (m-K_{i-1})/m * 1 + K_{i-1}/m * 0 = (m-K_{i-1})/m$$

$$where\ K_i = \Delta K_i + K_{i-1}$$

(Equation 1)

Hence from (Equation 1) the expected number of classes that will be acquired after case *n* will be the sum of all the classes acquired for that and all previously seen cases hence:

$$K_n = \Sigma_{(i = 1\ to\ n)} \Delta K_i = \Sigma_{(i = 1\ to\ n)} (m - K_{i-1})/m$$

$$= \Sigma_{(i = 1\ to\ n)} (1 - K_{i-1} / m)$$

$$= (\Sigma_{(i = 1\ to\ n)} 1) - (\Sigma_{(i = 1\ to\ n)} K_{i-1} / m)$$

therefore:

$$K_n = n - (1/m * \Sigma_{(i = 1\ to\ n)} K_{i-1})$$

(Equation 2)

QED. (Equation 2) results in Theorem 1.

7.2.4.2 Cases Seen as function of Classes Acquired

The aim of this section is to prove the following theorem:

Theorem 2: The expected number of cases required to acquire K of m classes in a single classification case-driven KA system is given by:

$$n_K = \Sigma_{(i = 1 \text{ to } K)} [(m-(i-1))/m) * \Sigma_{(p = 1 \text{ to } \infty)} \{p * ((i-1)/m)^{(p-1)}\}]$$

where:

n is the number of cases required;

K is the amount of classes to be accumulated; and

m is the total number of classes in the domain.

Theorem 2 provides a method for discovering the expected number of cases that would be required to acquire a portion of the *m* classes in a given equal frequency single classification domain. However its formula is computationally more expensive than the reciprocal formula in Theorem 1 since it involves a complex summation of an infinite number of terms. This reflects the asymptotic nature of the expected trajectory. Depending on the number of classes to be acquired *K*, for $K < m$ it is possible to trade-off the accuracy of the solution with the number of terms used in Theorem 2.

Proof 2: Proof of Theorem 2

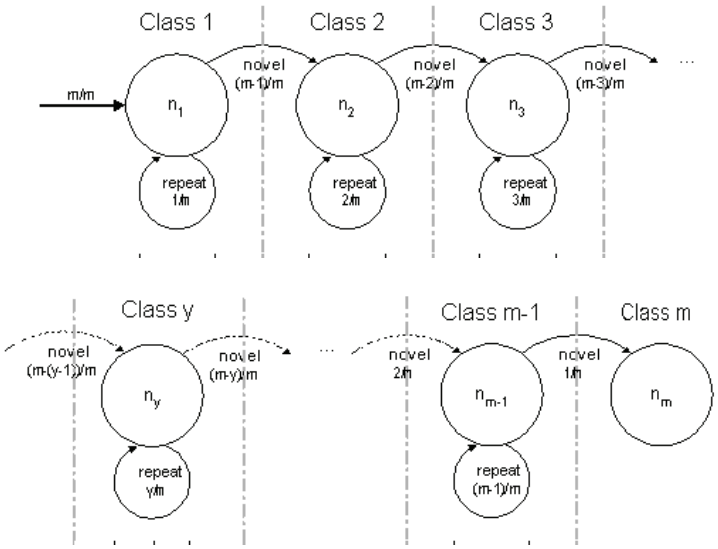
In order to determine the number of cases required to acquire a given number of unique classes, the KA state transitions were represented in a data flow diagram (DFD). In Figure 12 (page 106), the circles represent the state of knowing the indicated class and the paths show the probability of either returning to that state when a repeat exemplar is seen, or of moving to the next state when a novel exemplar is seen[124].

In a single classification case-driven KA system, there are different ways in which the KBS can transition from knowing Class (y-1) to knowing Class (y). When the next case is an exemplar for a novel class the KBS will transition directly from Class (y-1) to Class (y). But when the next and possibly subsequent cases are exemplars for known classes they appear as

[124] Anonymous reviewers of the Vazey(2006) EKAW submission and a submission to the SGAI AI-UK 2006 conference likened this work to a random walk or Markov chain. It has been left for future researchers to investigate this link.

repeats to the KBS. Any number of repeats could occur before the KBS transitions to the next knowledge state.

Figure 12: Acquiring Classes on the basis of Cases



Let the expected total number of cases *n* that needs to be seen before Class 1 is acquired be n_1 and the expected number of cases that needs to be seen to move from zero classes to Class 1 be Δn_1. In that case:

$$n_1 = \Delta n_1 = m/m = 1$$

In other words, we expect that only one case needs to be seen before Class 1 is acquired.

From Figure 12, the chance that only one case needs to be seen to move from Class 1 to Class 2 is given by: *(m-1)/m*. The chance that two cases will need to be seen to move from Class 1 to Class 2 is given by: *(m-1)/m * (1/m)* and so on.

Therefore the expected number of cases Δn_2 that needs to be seen to move from Class 1 to Class 2 will be given by the weighted sum of the probabilities of moving between Class 1 and Class 2 with zero or more repeat exemplar cases being seen between these two classes. Hence:

$$\Delta n_2 = 1 * (m-1)/m$$

$$+ 2 * (m-1)/m * (1/m)$$

$$+ 3 * (m-1)/m * (1/m)^2 \ldots$$

$$+ j * (m-1)/m * (1/m)^{(j-1)} \ldots \text{ and so on}$$

Where j represents the jth term in which (j-1) repeat exemplars are seen before the given class is acquired, hence:

$$\Delta n_2 = (m-1)/m * [1 + 2 * (1/m) + 3 * (1/m)^2 + \ldots + j * (1/m)^{(j-1)} + \ldots]$$

Note that the expected total number of cases n that needs to be seen before Class 2 is acquired will be $n_2 = \Delta n_2 + n_1$.

Continuing along these lines we get the expected number of cases that needs to be seen to move from Class 2 to Class 3 i.e. Δn_3:

$$\Delta n_3 = (m-2)/m * [1 + 2 * (2/m) + 3 * (2/m)^2 + \ldots + j * (2/m)^{(j-1)} + \ldots]$$

Where the expected total number of cases n that needs to be seen before Class 3 is acquired is

$$n_3 = \Delta n_3 + n_2$$

By induction we get:

$$\Delta n_i = (m-(i-1))/m) *$$

$$[1 + 2 * ((i-1)/m)$$

$$+ 3 * ((i-1)/m)^2 + \ldots$$

$$+ j * ((i-1)/m)^{(j-1)} + \ldots \text{ and so on}]$$

Hence:

$$\Delta n_i = (m-(i-1))/m) * \Sigma_{(p = 1 \text{ to } \infty)}\, p * ((i-1)/m)^{(p-1)}$$

$$\text{where } n_i = \Delta n_i + n_{i-1}$$

(Equation 3)

The number of classes that will be acquired after case n will be the sum of all the classes acquired for that and all previously seen cases. Hence from (Equation 3) the expected number of cases that needs to be seen to acquire K of m classes in total is:

$$n_K = \Sigma_{(i = 1 \text{ to } K)}\, \Delta n_i$$

therefore:

$$n_K = \Sigma_{(i = 1 \text{ to } K)} [(m-(i-1))/m) * \Sigma_{(p = 1 \text{ to } \infty)} \{p * ((i-1)/m)^{(p-1)}\}]$$

(Equation 4)

QED. (Equation 4) results in Theorem 2.

7.2.4.3 Distribution about the Expected KA Trajectory

For the sake of discussion, after some number $y < m$ of m classes have been acquired in a KA system, two things can happen:

1. A case that represents a novel class is received and the KA process achieves the $(y + 1)$ th class; or
2. Some number Δx of cases that each map to some existing class in the system are received . Such cases are referred to here as *repeats*.

For example, after the 1st case is seen, the 1st class is acquired. Then $\Delta x_1 = 0$ or more repeat cases will occur, until the 2nd class is acquired, then $\Delta x_2 = 0$ or more repeat cases will occur, until the 3rd class is acquired, and so on... The total number of repeat cases seen so far will be the sum of all the repeat cases experienced in-between the novel cases and hence class acquisitions.

In order to examine the probabilistic distribution of the number of cases that need to be seen to acquire a given number of classes, the matrix in Table 9 (page 109) shows the probability of being at a given class y after a given number of repeat cases x for an $m = 10$ class system. The matrix is constructed by recognizing that the probability P_{yx} of having acquired y classes having seen x repeat cases is given by:

1. the probability $P_{(y-1)(x)}$ of having acquired only $(y-1)$ classes after x repeats over the duration of the KA task, followed by (and hence multiplied by) the probability P_{y0} of achieving the yth class on the next case; plus
2. the probability $P_{y(x-1)}$ of having acquired y classes with $(x-1)$ repeats followed by (and hence multiplied by) the probability P_{01} of yet another repeat being received at class y. This can be expressed as follows:

$$P_{yx} = P_{(y-1)(x)} * P_{y0} + P_{y(x-1)} * P_{01}$$

Note from Figure 12 on page 106 that the probability P_{y0} of achieving the *y*th class is given by *(m-(y-1))/m*. Also from this figure, the probability P_{01} of yet another repeat is given by *y/m*. Hence:

$$P_{yx} = P_{(y-1)(x)} * (m-(y-1))/m + P_{y(x-1)} * y/m$$

(Equation 5)

Table 9: An example KA probability matrix for a 10 class system

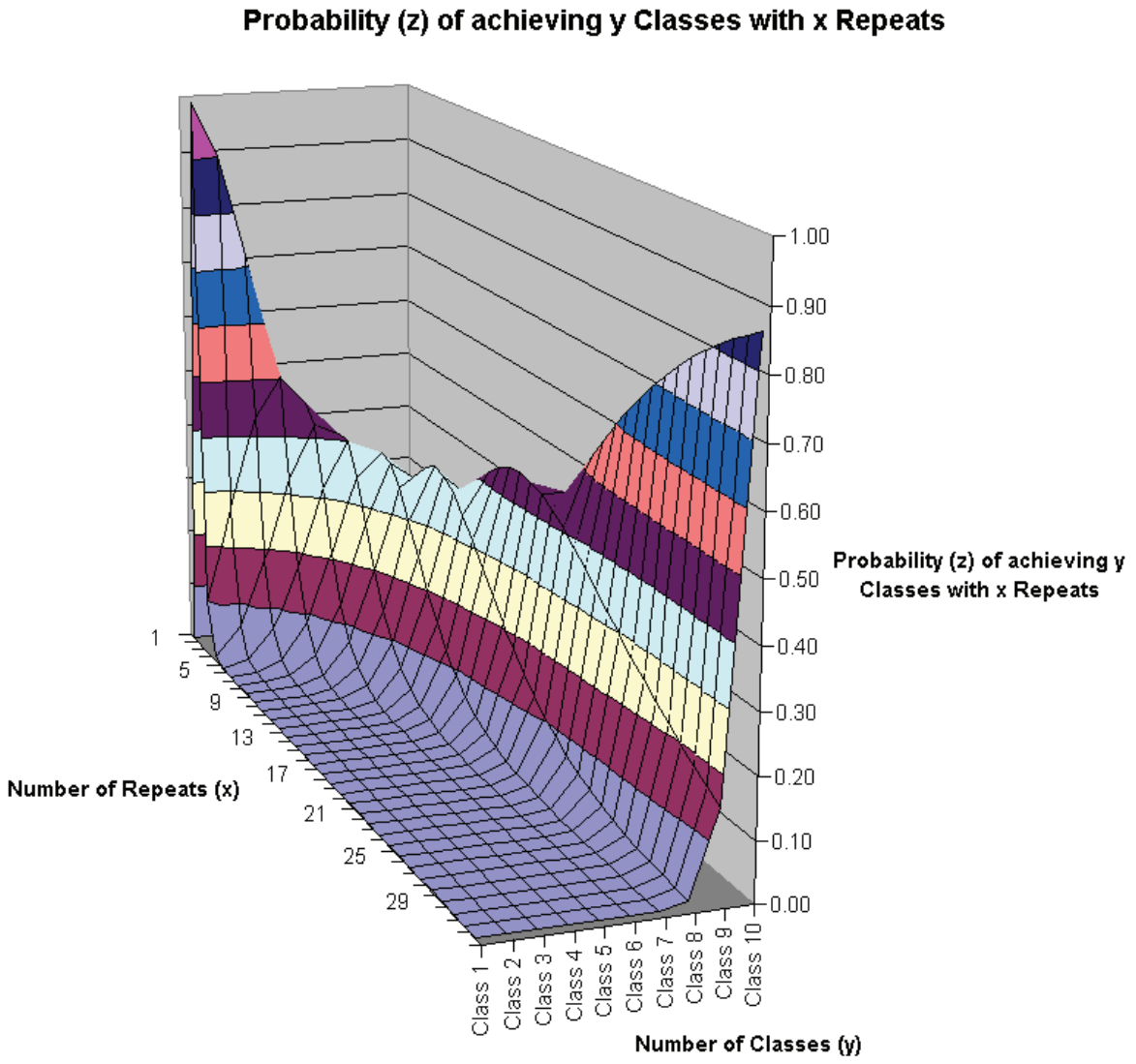
repeats (x) \ classes (y)	1	2	3	4	5	6	7	8	9	10
0	1.00	0.90	0.72	0.50	0.30	0.15	0.06	0.02	0.00	0.00
1	0.10	0.27	0.43	0.50	0.45	0.32	0.17	0.07	0.02	0.00
2	0.01	0.06	0.18	0.33	0.42	0.40	0.28	0.14	0.04	0.01
3	0.00	0.01	0.06	0.18	0.32	0.40	0.36	0.22	0.08	0.01
4	0.00	0.00	0.02	0.09	0.21	0.35	**0.39**	0.29	0.13	0.03
5	0.00	0.00	0.01	0.04	0.13	0.27	0.38	0.34	0.19	0.05
6	0.00	0.00	0.00	0.02	0.07	0.20	0.35	0.38	0.24	0.07
7	0.00	0.00	0.00	0.01	0.04	0.14	0.30	0.39	0.30	0.10
8	0.00	0.00	0.00	0.00	0.02	0.10	0.25	0.39	0.35	0.13
9	0.00	0.00	0.00	0.00	0.01	0.06	0.20	0.37	0.39	0.17
10	0.00	0.00	0.00	0.00	0.01	0.04	0.16	0.34	0.42	0.21
11	0.00	0.00	0.00	0.00	0.00	0.03	0.12	0.31	0.44	0.26
12	0.00	0.00	0.00	0.00	0.00	0.02	0.09	0.28	0.45	0.30
13	0.00	0.00	0.00	0.00	0.00	0.01	0.07	0.24	0.45	0.35
14	0.00	0.00	0.00	0.00	0.00	0.01	0.05	0.21	0.45	0.39
15	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.18	0.44	0.44
16	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.15	0.42	0.48
17	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.13	0.41	0.52
18	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.10	0.39	0.56
19	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.09	0.37	0.59
20	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.07	0.34	0.63

Please note: This table has been truncated at 20 repetitions.

This matrix shows the distribution of actual KA trajectories about the expected trajectory for an *m = 10* class system for the situation where repeat exemplars are drawn up to 20 times between each class. Figure 13 on page 110 graphs the distribution of KA probabilities for Table 9. From the table, we can see that for a 10 class system, the probability of achieving 7 classes with only 4 repeats is 0.39 which is the same as the probability of achieving 6 classes

with only 4 repeats (0.35) multiplied by the probability of achieving the 7th class (10-(7-1))/10 plus the probability of achieving 7 classes with only 3 repeats (0.36) multiplied by the probability of yet another repeat 7/10.

Figure 13: An example KA probability graph for a 10 class system



From Table 9 and Figure 13 it becomes clear that the probability of repeats is relatively low and almost zero in the early stages of KA (only two decimal places are shown but the values are actually non-zero), but as more classes are achieved, the probability of repeats and the variance of the number of repeats becomes much higher. In the example provided, for the 5th through to the 9th class acquisition, the probability of repeats peaks and then decreases with the number of repeats seen. For the final class acquisition, the probability of repeats actually

increases, and the variance in the number of cases required to achieve the final class is infinite, reflecting the asymptotic nature of case-driven KA.

An interesting property of the theoretical matrix that one can create is that the probabilities on the "diagonals"[125] add to 1. For example if you have seen 3 cases, you've either seen 1 novel exemplar and 2 repeats, 2 novel exemplars and 1 repeat, or 3 novel exemplars. (Please note that the matrix in Table 9 has been truncated at 20 repetitions simply so that it can fit on the page.)

Table 9, and Figure 13 also indicate that the number of terms required for a useful level of solution accuracy in Theorem 2 increases significantly with the number of classes acquired. This can be observed in Table 9 by noticing for instance that at two decimal places, to estimate the number of cases needed to acquire 3 classes requires that at least 5 repeat cases be considered, but to estimate the number of cases needed to acquire 6 classes requires that at least 14 repeat cases be considered.

7.2.4.4 Likelihood of the Best Case

Note that in an optimal (manual or machine learning) KA scenario one could argue that all of the knowledge would be acquired up front[126]. In that way the knowledge base would be prepared for all future scenarios in advance, and in the case of manual KA, experts wouldn't have to put up with the tedious review of repeat exemplar cases while offering their knowledge for novel exemplar cases. By this logic, each new case would optimally represent a unique class and the best-case KA trajectory would be a straight line as shown in Figure 11 on page 102[127].

---

[125] In this context the term "diagonal" is taken to mean moving right by one cell and up by one cell across the matrix.

[126] As well, optimally the knowledge base would acquire the most useful knowledge first e.g. knowledge that is most likely to be reused, and which will provide the most useful advice. We return to this point in section 7.2.6 on page 118.

[127] Although repetitive review of repeat exemplar cases for the same class can be seen as non-optimal in terms of the amount of user interaction required, where cases are repetitive, statistics can be gathered as to the number of times a particular class is represented and these statistics may help to improve the credibility of the acquired class data.

From Figure 12 on page 106, the likelihood of achieving the best case (shortest path) KA trajectory using a case-driven KA method is given by:

$$P_{BestCase} = m! / m\text{^}m$$

(Equation 6)[128]

Hence for large *m* the best case KA outcome is extremely rare[129].

7.2.5 Generalising to Multiple Classification KA systems

Where cases are mapped to more than one class in a case-driven KA system, it is possible that more than one class will be acquired for a given exemplar case.

In order to examine the multiple-classification case-driven KA process 3 sets of data were created, each with $N = 1000$ randomly generated cases. Cases in the first set of data were each mapped to one of $m = 100$ different classes. Cases in the second set of data were each mapped to two of $m = 100$ different classes. Cases in the third set of data were each mapped to three of $m = 100$ different classes. The *m* different classes were randomly distributed and occurred with equal frequency in each example. Note that for multiple classification case-driven KA as in MCRDR systems, each novel class represents a new RuleNode in the decision tree.

The number of novel classes versus the number of cases seen was cumulatively counted and plotted. Table 10 (page 113) illustrates the example. For a 1 class system, only novel classes in the first column were counted. For a 2 class system, novel classes in the first two columns were counted. Finally, for a 3 class system, novel classes in all 3 columns were counted. The cumulative number of novel classes seen in each of the 3 different KA systems is shown in the last three columns of the table. Figure 14 (page 114) shows the expected and actual trajectories for a KA system with $m = 100$ classes and $N = 1000$ cases in which 1, 2, and 3 classes are mapped to each case.

---

[128] Note that in (Equation 6) ! means *factorial* i.e. m! = [m * (m-1) * (m-2) *…* 2 * 1] and ^ means *to the power of*.

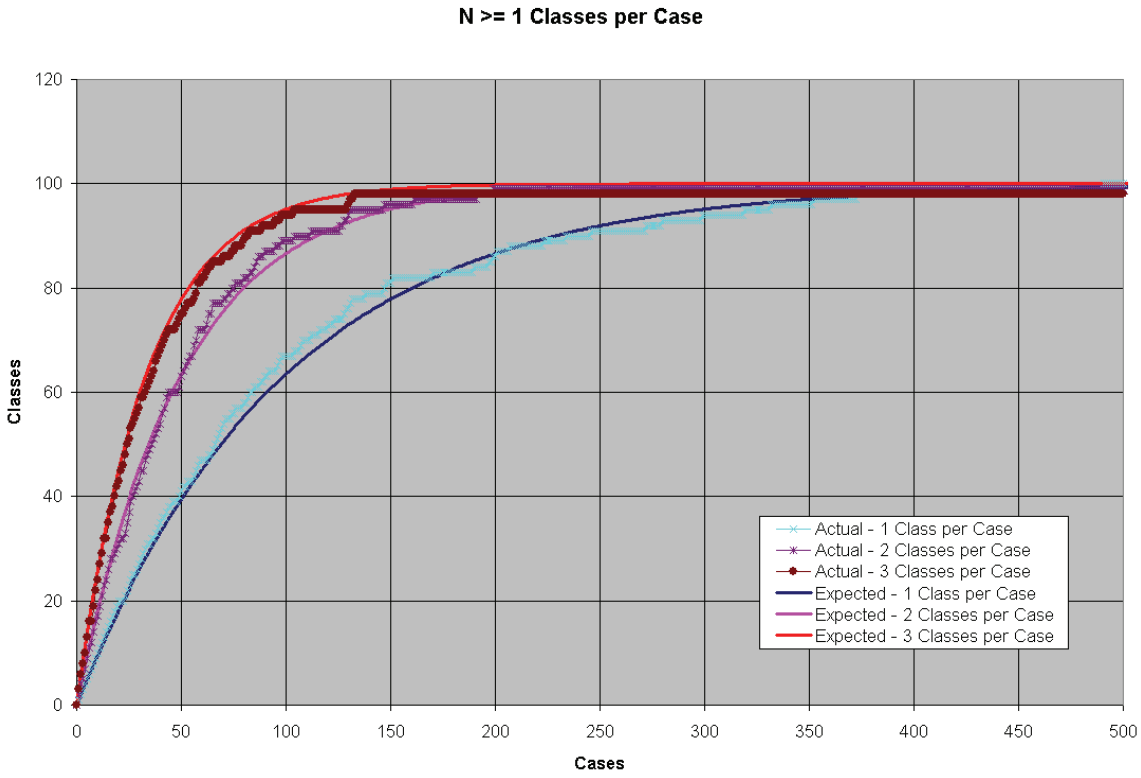
[129] An anonymous reviewer of the Vazey(2006) EKAW submission suggested that users may not be really interested in reaching the asymptote itself, but getting there sufficiently closely (say 90%, reminiscent of Simon's satisficing notion). It is left to future researchers to investigate this further.

Table 10: Cumulative count of novel classes acquired in a Multiple Classification KA study

(Note: Data for the first 12 cases is displayed for a 100-class KA domain)

Case ID (1-1000)	First Random class ID (1 – 100)	Second Random class ID (1 – 100)	Third Random class ID (1 – 100)	1 class system Cumulative count of novel classes acquired (<=100)	2 class system Cumulative count of novel classes acquired (<=100)	3 class system Cumulative count of novel classes acquired (<=100)
1	87	2	7	1	2	3
2	8	32	76	2	4	6
3	41	39	59	3	6	9
4	21	64	1	4	8	12
5	15	79	32	5	10	14
6	35	42	24	6	12	17
7	35	84	80	6	13	19
8	46	17	26	7	15	22
9	61	19	61	8	17	23
10	7	38	60	9	19	25
11	56	88	10	10	21	28
12	76	4	14	11	23	30
…	…	…	…			…

Figure 14: Expected and Actual Multi-Class Case-Driven KA trajectories.



7.2.5.1 Classes Acquired as function of Cases Seen

The aim of this section is to prove the following theorem:

Theorem 3: The expected number of classes K_n that will be acquired after n cases in a case-driven KA system where c classes are acquired per case is given by:

$$K_n = K_{n-1/c} + (m-K_{n-1/c})/m$$

where:

n is the number of cases seen;

K_n is the number of classes accumulated after n cases and $K_0 = 0$ and $K_{1/c} = 1$;

c is the number of classes acquired per case; and

m is the total number of classes in the domain.

This theorem was used to construct the expected equal frequency multiple classification KA trajectories shown in the Figure 14 on page 114.

Proof 3: Proof of Theorem 3

From (Equation 1, page 104), the probabilistically expected number of classes K_n that will be acquired after n cases by a single classification case-driven KBS is given by:

$$K_i = K_{i-1} + \Delta K_i$$

where:

$$\Delta K_i = (m-K_{i-1})/m * 1 + K_{i-1}/m * 0 = (m-K_{i-1}) / m$$

hence:

$$K_i = K_{i-1} + (m-K_{i-1}) / m$$

(Equation 7)

By analogy, in a domain in which the number of classifications obtained per case is *c*, we notice that every *(1/c)* cases, the expected amount of knowledge obtained is:

$$\Delta K_{(i/c)} = (m - K_{(i/c) - (1/c)}) / m$$

Continuing this analogy, in generalising (Equation 7) we get:

$$K_{(i/c)} = K_{(i/c) - (1/c)} + \Delta K_{(i/c)}$$

$$K_{(i/c)} = K_{(i/c) - (1/c)} + (m - K_{(i/c) - (1/c)}) / m$$

and hence:

$$K_n = K_{n - (1/c)} + (m-K_{n - (1/c)}) / m$$

(Equation 8)

QED. (Equation 8) results in Theorem 3.

As an example, let's apply Theorem 3 to a domain where two classifications are obtained per case. In that case, we get:

$$K_n = K_{n-1/2} + (m-K_{n-1/2})/m \quad \text{and}$$

$$K_{n-1/2} = K_{n-1} + (m-K_{n-1})/m$$

Hence, the probabilistically expected number of classes K_n that will be acquired after n cases by a bi-classification case-driven KBS is given by:

$$K_n = K_{n-1} + (m-K_{n-1})/m + (m-(K_{n-1}+(m-K_{n-1})/m))/m$$

(Equation 9)

This equation was used to graph the expected trajectory for the bi-classification domain in Figure 14 and it provides a very good correlation with the corresponding experimental data shown in the figure.

In a different example, applying Theorem 3 to a domain where three classifications are obtained per case gives:

$$K_n = K_{n-1/3} + (m-K_{n-1/3})/m \quad \text{and}$$

$$K_{n-1/3} = K_{n-2/3} + (m-K_{n-2/3})/m \quad \text{and}$$

$$K_{n-2/3} = K_{n-1} + (m-K_{n-1})/m$$

Hence the probabilistically expected number of classes K_n that will be acquired after n cases by a tri-classification case-driven KBS is given by:

$$K_n = K_{n-1} + (m-K_{n-1})/m + (m-(K_{n-1}+(m-K_{n-1})/m))/m +$$

$$(m-(K_{n-1} + (m-K_{n-1})/m + (m-(K_{n-1}+(m-K_{n-1})/m))/m))/m$$

(Equation 10)

This equation was used to graph the expected trajectory for the tri-classification domain in Figure 14 and it provides a very good correlation with the corresponding experimental data shown in the figure.

7.2.5.2 Cases Seen as function of Classes Acquired

Since more than one class may occur for each exemplar case in a multiple classification KA system, we can expect that its case-driven KA trajectory would rise faster than in an single classification KA system i.e. less cases need to be seen before the same level of knowledge is acquired. As indicated by the following theorem, it turns out that this corresponds to a linear shrinking of the x-axis in proportion to the number of classes acquired per case.

Theorem 4: The expected number of cases required to acquire K of m classes in a case-driven KA system where c classes are acquired per case is given by:

$$n_K = 1/c \sum_{(i = 1 \text{ to } K)} [(m-(i-1))/m) * \sum_{(p = 1 \text{ to } \infty)} \{p * ((i-1)/m)^{(p-1)}\}]$$

where:

n is the number of cases required;

K is the amount of classes to be accumulated;

c is the number of classes acquired per case; and

m is the total number of classes in the domain.

For example, in Figure 14, 80% of the classes are achieved after 158 cases for a uni-classification system, after 79 cases for a bi- classification system (2 * 79 = 158), and after 53 whole cases for a tri-classification system (3 * 53 = 159).

The simulations conducted for the Chess, TicTacToe and Garvan[130] domains (Kang, 1995, pp 94 – 95, 105-107), it was observed that the variance of the output at all levels of expertise was greater for SCRDR than for MCRDR (pp 95, 109, 127). At that time, it was conjectured that the use of multiple cornerstone cases in MCRDR, and the ability for users to chose the location of MCRDR rules so that they were less likely to be hidden away in local contexts, produced a more robust system less affected by the order of cases. This research indicates that one of the reasons[131] why there is a difference in variability is because MCRDR can

[130] Note that even though the chess and TicTacToe domains were only bi-conclusion domains, new cases could still fetch multiple classifications. This is clearly demonstrated in (Kang, 1995, p 108, p115) where the number of cornerstone cases included in the difference list and hence the number of dependent RuleNodes per RuleNode was up to 12, and 10 respectively for the Chess and Tic-Tac-Toe domains. For these bi-conclusion domains, the fetching of multiple classifications could be due to overlapping sibling RuleNodes offering identical conclusions. See also (Kang, 1995, p 65). In section 8.3.10 on page 155, we also note that very little depth in the rule tree was acquired in these two test domains.

[131] This is just one of many other reasons why there is a difference in variability between Kang;s SCRDR and MCRDR results. Some of the other factors are discussed in section 7.3.1 (page 124) and section 7.6 (page 129) of the thesis.

acquire multiple classes for each case seen, and hence it needs to see fewer cases, with lesser overall variance (in linear proportion to the number of classes acquired per case), to acquire the same number of novel classes.

Note that in a real-life multiple classification scenarios, certain combinations of classes may be more likely to co-occur than others. The next section offeres a discussion for classes that occur with different frequencies in single classification domains.

7.2.6 Classes occurring with different frequencies

In many domains, including the ICT support centre domain, the classes being acquired do not occur with equal frequency across the cases seen. For example, at the support centre, 60% of the problems may occur because of problem type A, 30% because of problem type B, 5% because of problem type C and so on, with the remaining 5% being of various different types.

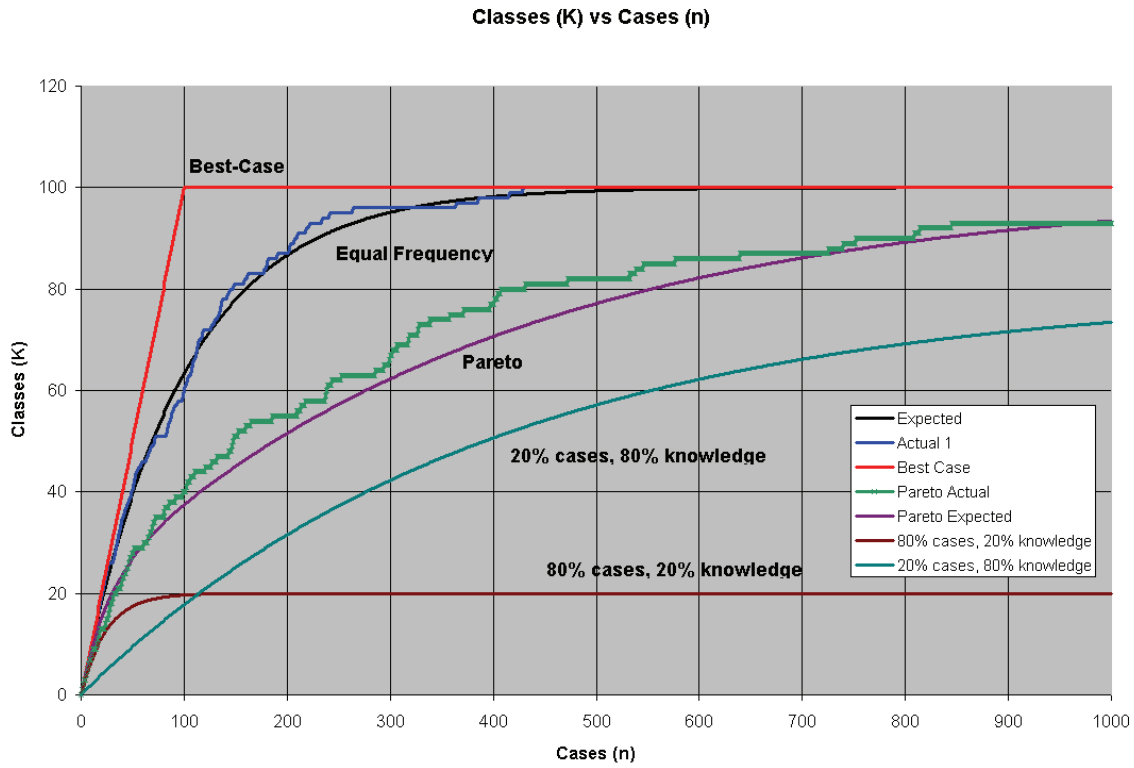
A further example is provided by the PEIRS SCRDR data. After 25 months of PEIRS operation, 37% of the interpretations were for thyroid function tests (TFTs), 43% for arterial blood gases (ABGs), and 20% for other tests (Edwards, 1996, p97). At completion of the trial, the number of RuleNodes acquired were 33% for TFTs, 49% for ABGs and 18% for other tests (p101). So the ratio of TFT Cases to TFT RuleNodes was more than for ABG Cases to ABG RuleNodes, reflecting that the multiple classification ABG domain was more complex to translate into the SCRDR data structure than the simpler single classification TFT domain. This is further supported by Edwards (pp 104 - 105).

To model the situation where the classes being acquired do not occur with equal frequency across the cases seen N = 1000 cases were constructed each comprising one of m = 100 different classes. The first m_1 = 20% of the m different classes were represented in r_1 = 80% of the cases. The last m_2 = 80% of the m different classes were represented in only r_2 = 20% of the cases. This modelled the well-known Pareto[132] 80/20 distribution. As noted in section 5.4 (page 72), tags used in broad folksomonies also tend to follow the Pareto power-law distribution, as does the popularity of web pages more generally. The cases were sorted so that the order of classes represented by them was completely random.

[132] http://en.wikipedia.org/wiki/Pareto_principle

Again, a cumulative count was made of the number of times a case with a novel integer was seen and the number of novel classes seen versus the number of cases seen was plotted. Figure 15 shows the Best Case, Actual and Expected trajectories of the randomly generated case data for m = 100 for the single classification equal frequency class example and for this unequal (Pareto) frequency class example. The component parts of the expected Pareto trajectory are also shown. From this figure we can see that the total rate of knowledge acquisition for a domain with a Pareto style distribution is much slower than for an equal frequency problem domain.

Figure 15: An example showing KA for classes with both equal and different frequencies



As we see in Figure 15, although the total KA is much slower for a Pareto-style distribution, the most frequently demanded knowledge is generally acquired first, and the least frequently demanded knowledge is generally acquired last. Hence the variation of frequencies between the classes acquired is unlikely to reduce the effectiveness of case-driven KA as a KA mechanism. In fact, in many domains knowledge acquired for the highest volume problems has the best potential to save on labour costs and is therefore of top priority. As noted by

(Beydoun et. al, 2005, p52), "*the real performance of the conceptual model depends on instances with frequent occurrence, more than instances of rare occurrence*".

As shown in the following theorem, the formula for the expected trajectory for data that occurs with different frequencies is given by summing (Equation 1) across each of the different sub-domains represented by the data, in proportion to the actual knowledge gain achieved in acquiring a class in that sub-domain.

Theorem 5: The expected number of classes K_n that will be acquired after n cases in a case-driven KA system with knowledge sub-domains M_1 through to M_j is given by:

$$K_n = K_{n-1} + \Delta K_n \qquad \textit{where}$$

$$\Delta K_n = \sum_{1 \text{ to } j} (m_j - K_{j,n-1})/m_j * r_j \quad \textit{and}$$

where:

- n is the number of cases seen;
- $K_{j,n}$ is the number of classes accumulated in sub-domain M_j after n cases, and if M_j is a single classification domain, $K_{j,0} = r_j$;
- m_j is the total number of classes in sub-domain M_j.

So in the dual sub-domain example, we get:

$$\Delta K_i = (20 - K_{1,i-1})/20 * 0.8 + (80 - K_{2,i-1})/80 * 0.2$$

$$\text{where } K_{1,0} = 0.8 \text{ and } K_{2,0} = 0.2$$

Note that unequal frequency classes affect the rate at which accuracy can be achieved in the KBS. Figure 15 shows that in the Pareto example, 100% accuracy for 80% cases was achieved by about 150 cases. However, since these cases only represent 20% of the possible classes i.e. RuleNodes, the total knowledge acquired by the system was still only around 50% at that point in time.

It has previously been conjectured that as the traditional knowledge bases grow, the *knowledge acquisition bottleneck*[133] worsens because of the inherent difficulty knowledge engineers have in dealing with such systems. Data from the GARVAN-ES1 system has often been given as an example of this *knowledge acquisition bottleneck* (Compton and Jansen, 1989) (Edwards, 1996, p51) and used to justify an SCRDR or MCRDR approach. For example, in the Garvan-ES1 introduced in 1984 and reported by Compton (Compton and Jansen, 1989, p6) and Kang (1995, p15) a doubling in size of the knowledge base was required to take the accuracy from 96.4% to 99.7% (see also Kang, Gambetta, Compton, 1996, p258).

But what if the GARVAN-ES1 cases displayed an uneven distribution of classes as modelled here? In fact, Kang, Gambetta and Compton (1996, p263) report that 77.3% of GARVAN-ES1 cases had the default classification, indicating a very uneven class distribution indeed.

In Figure 15, after ~150 cases have been seen by the system, and after just 40 classes have been acquired, 20/20 = 100% accuracy is achieved for 80% of the cases, whereas only ~25/80 = 31.25% accuracy is achieved for 20% of the cases. This gives a weighted average accuracy of (1 * 0.8 + 0.3125 * 0.2) = 86% after just 45 RuleNodes. In this example, due to the infrequency of 20% of the classes, the knowledge base must more than double in size to achieve the last 14% accuracy at 100 RuleNodes, irrespective of whether a SCRDR, MCRDR or traditional Expert System structure is used.

From this analysis, the rate of accumulation of classes or RuleNodes, and hence the rate of improvement in system accuracy may have more to do with the probabilistic properties of random and unevenly represented classes, than it has to do with the particular case-driven knowledge acquisition paradigm used.

7.2.7 Multiple Parties Transferring Knowledge

If multiple parties are randomly transferring knowledge to the KBS we can model their contribution as a round-robin contribution on the basis of cases seen. In that case the combined KA trajectory for all users will corroborate the expected trajectories presented thus far, but for each individual user, the knowledge acquired as a function of the cases seen by

[133] The *knowledge acquisition* bottleneck is a term coined by Feigenbaum (1980) as discussed in (Edwards, 1996, p50). Further discussion is provided in section 8.3.4 (page 145).

that particular user will be much more rapid since other users are contributing to the same KBS and the user will need to see fewer cases for the same amount of knowledge gain.

More specifically, the expected incremental amount of knowledge ΔK_i that will be acquired for a given user after each case will be affected by total amount of knowledge contributed by all users up until that point K_{i-1} where i is the total number of cases seen by the whole KBS rather than just by that individual. If users contribute classes that occur with different frequencies we can extend the Pareto analysis provided earlier to model their individual and combined contributions.

7.2.8 Sticky Knowledge

The analysis thus far has assumed that knowledge is sticky i.e. once it has been transferred and received it is also stored and the KA process will not work to deplete the knowledge accumulated thus far. This assumption may or may not hold depending on the particular KA scenario.

7.2.9 Remaining Error

Let $K_{Error\ n}$ refer to the difference between the amount of knowledge K_n acquired after n cases, and the maximum amount of knowledge that could be acquired m. Hence the remaining error $K_{Error\ n}$ is as follows:

$$K_{Error\ n} = m - K_n$$

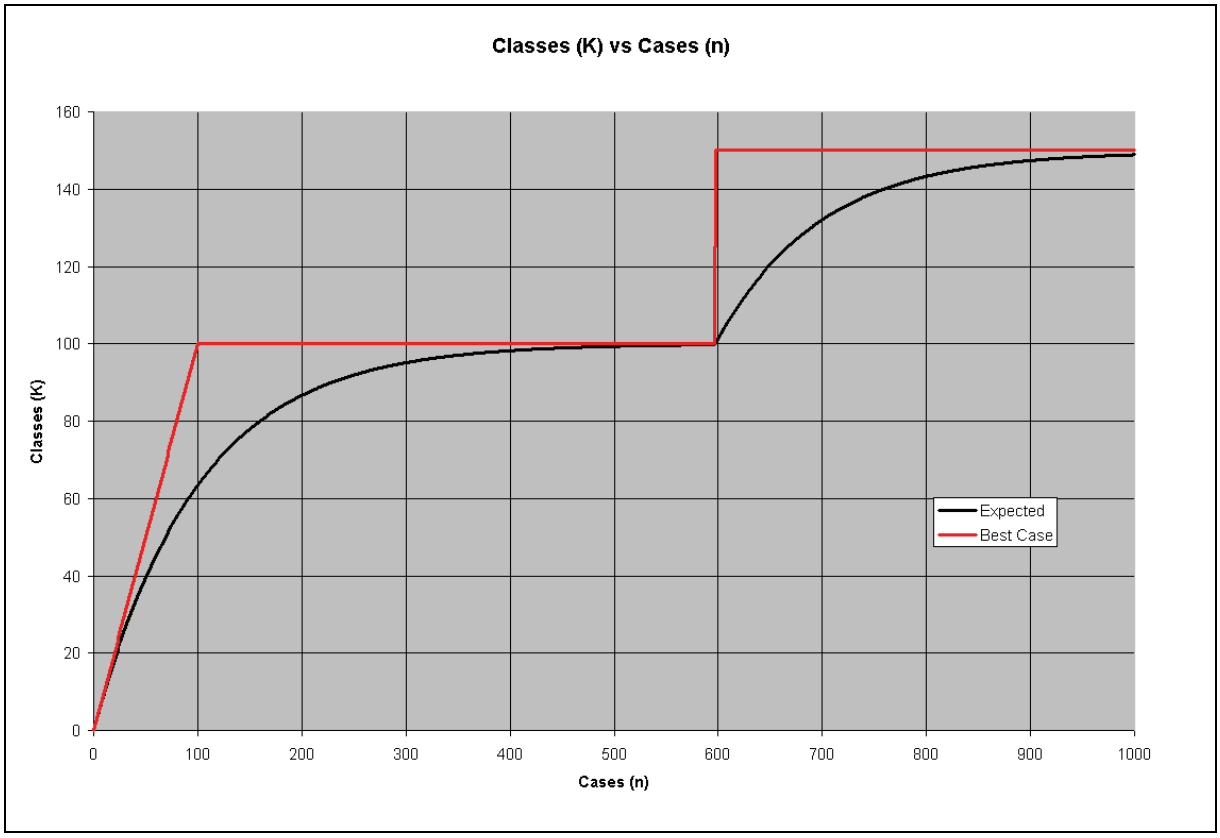
(Equation 11)

where K_n is given for example by Theorem 1, Theorem 3, or Equation 11.

7.2.10 Inclusion of New Sub-domains

Note that as new sub-domains are included to the KBS, the total number of classes (m) that may be acquired will be increased, and the shape of the case-driven trajectory will change to reflect the additional knowledge. This is demonstrated in Figure 16 (page 123) after case (n = 600).

Figure 16: Inclusion of a new sub-domain



7.2.11 Upward drift in the domain size

An upward drift in the size of the domain may also occur for example, if a user's demand for specificity versus generality in the classifications increases. This refinement effect may be the factor contributing to the upward drift of the case-driven Folksomony KA trajectories presented in Figure 9 (page 96). Another example in the upward drift of knowledge that can be acquired and used in a domain is given in (Beydoun et. al 2007, p8) and is attributed to users learning from each-other as the system captures and conveys more knowledge. Dazeley and Kang (2004, p4) also note the importance of capturing inter-RuleNode relationships. The reflexive nature of learning is discussed further in section 7.5 (page 128).

7.2.12 Imperfect knowledge

For the interested reader, a discussion of how the derived KA models change when the knowledge being transferred is false, speculative or unknown is provided in Appendix Q on page 473. In that appendix, reference is also made to the work of Compton (2000), and (Cao and Compton, 2005, 2006) in simulating the affect of overgeneralisation and

overspecialisation errors on SCRDR, Flat RDR and Composite Rule KA trajectories in the presence or absence of cornerstone cases and taxonomic hierarchy. Cao and Compton's simulation work was also previously discussed in section 8.3.8 (page 151) and 8.3.9 (page 153).

7.3 Review of Past Experimental Data

7.3.1 Simulated Expertise

Three different types of simulated experts were used by Compton, Preston and Kang (1995) to evaluate the transfer of knowledge by a simulated expert from a machine learnt KBS to an SCRDR KBS. The smartest expert (S1) made an informed decision about the rules it should add to the SCRDR decision tree based on the intersection of a difference list generated by comparing the wrongly classified case with the case that was initially used to create the incumbent RuleNode, and up to 4 of the conditions derived in the machine-learnt knowledge-base[134]. In contrast, the moderately smart expert (S2) took an over-generalization bias leading to excessive false-positives, and the dumbest expert (D) took an over-specialization bias leading to excessive false-negatives.

The researchers found that S1 produced the smallest knowledge bases, whereas S2 and D produced the largest knowledge bases. The model presented herein readily explains these results since the amount of knowledge encoded for any one case by the smartest expert S1 was at least 4 times that of the moderate expert S2. This corresponds to a fewer classes in the domain (m) for the smartest expert S1 as compared to the moderately smart expert S2. As well, the dumbest expert D was so biased to over-specialization that a separate class could have been created for every case with a unique combination of attributes.

[134] Kang, Compton and Preston (1995) found that for the smartest expert, selecting 4 conditions gave the entire intersection of a classification in nearly all cases and on average, only 2-3 rules needed to be satisfied by a case in order to reach the correct solution. Further, Compton (2005) refers to experience with SCRDR systems (Compton and Edwards et al 1991) in PEIRS where the average depth of the decision tree is 2-3 rules and the maximum is 6[134]. A corresponding statement for search engines would be that most people can find what they want using 2-3 keywords, with a maximum of 4-6 depending on the nature of the search domain. Naturally this result is very data dependent, but for a number of domains, for example in general when searching Google, it seems to be a good rule of thumb.

This research shows that the differences in the trajectories observed across the different domains, experts and machine-learnt knowledge bases in Compton, Preston and Kang (1995) and Kang et al. (1998) can be explained by:

- the chosen machine learning algorithm, and its over- or under-specialisation bias (e.g. S1, S2 or D) which affects the number of possible classes *m* in the domain;
- the preference for generality or specificity in the classes applied;
- the type of frequency distribution of the represented classes;
- the number of different RuleNodes / classifications / classes acquired per case;
- the amount of errors in the case attributes and the classifications in the raw data that may result in extra (wrong) classes and hence a larger number of possible classes *m* in the domain; and
- the number of cases available to the learning algorithm (*n*).

7.3.2 MCRDR Performance

Kang, Compton and Preston (1995) evaluated the performance of the MCRDR algorithm in the pathology domain. The basic findings were that as the expert system sees more and more cases over its lifetime:

- the rate in which new rules need to be added to the system rapidly declines,
- the rate at which conclusions presented to the user need refinement rapidly declines,
- the corresponding error rate in which wrong conclusions (solutions) are presented to users drastically declines, and
- the corresponding rate at which correct solutions are presented to users rapidly increases.

The case-driven KA model provided herein predicts that each of the above observations will occur simply on the basis of case-driven KA via an incoming stream of randomised repetitive cases. No underlying KBS data structure is assumed.

Perhaps a more interesting result was that over the lifetime of the MCRDR system, the number of cases that users needed to compare the current case with when formulating a new rule and conclusion flattened out and thereafter remained steady (Kang, 1995, pp 111-115).

This research hypothesises that the case-driven acquisition of cornerstone cases per RuleNode in conventional MCRDR systems also follows the probabilistically expected case-driven KA trajectory derived herein.

The reason is that in conventional MCRDR systems, every dependent RuleNode has just one cornerstone case, and the cornerstone case list for the parent RuleNode is comprised of the cornerstone cases of all its dependent RuleNodes. Just as the acquisition of RuleNodes in the entire rule tree in conventional MCRDR systems follows the trajectory predicted by Theorem 1, the acquisition of RuleNodes in any sub-tree and hence sub-domain in conventional MCRDR systems will also follow the trajectory predicted by Theorem 1. Given the one-to-one correspondence between the cornerstone case and its RuleNode, the number of cornerstone cases in the derived cornerstone case list that users need to compare the current case with when formulating a new rule and conclusion will also follow the same trajectory, flattening out as the knowledge domain is mapped, and then remaining steady.

7.3.3 Merging separately created NRDR conceptual frameworks

As mentioned in section 5.2 (page 66), during the course of this thesis, Beydoun et. al (March 2005) studied the merging of separately created NRDR KBSs from independently operating experts that modelled identical domains. In Fig. 3.3 (Beydoun et. al, 2005, p53) present a graph that is aimed at representing the coverage(r) i.e. the probability that a randomly drawn case will be properly classified by a given KBS as a function of the number of classes in that KBS, where the KBS is built by integrating a series of smaller KBSs built by independent experts representing the same problem domain. In Fig 3.4 (Beydoun et. al, 2005, p53) present a corresponding graph that is aimed at representing the probability of inconsistencies in a given domain. In contrast, this chapter indicates that the coverage and error depends on the number of classes in the domain (m) together with the number of classes already acquired (K_n). Further to these graphs, the threshold tolerance ε used in theorems 4.1 and 6.1 of (Beydoun et. al, 2005, p53) appears to be modelled as a constant. In contrast, this research finds that a KBS can benefit from contributions by multiple experts, regardless of the magnitude of their differences. Multiple perspectives can co-exist in the 7Cs system proposed by this research (described later in Chapter 9 on page 161 and beyond).

7.3.4 Hypothesis Testing

The Kolmogorov-Smirnov (K-S) one-sample test allows a test of goodness of fit to be made between a set of independent observed scores and some specified theoretical distribution (Siegel, 1956 p47). It determines whether the scores in the sample can reasonably be thought to have come from a population having the theoretical distribution[135].

In the case-driven KA scenario the probability of subsequent cases representing a KA opportunity depends on the cases that have previously been seen and consequently the cumulative classes acquired. Hence in the case-driven KA scenario samples are not independent of each other. Personal communication with Prof. Malcolm Hudson and Prof. Jun Ma in the School of Statistics at Macquarie University indicates that a modified Kolmogorov-Smirnov (K-S) one-sample test may be useful in determining whether observed case-driven KA data fits the "gold standard" trajectories derived herein.

Unfortunately I was unable to obtain access to the raw data from the machine learnt SCRDR and MCRDR studies referred to herein. As well, my research has been focussed more on human-driven KA rather than machine-driven KA. Hence this task has been left for future researchers.

7.4 An Analysis of Rule-driven (RD) Knowledge Acquisition

As might be reasonably expected, top-down rule-driven KA (RDKA) forms a completely different knowledge acquisition trajectory than bottom-up case-driven KA. Top-down rule-driven KA refers to the situation (as in Conventional Expert Systems discussed previously in section 3.5.1 on page 36) in which users can add RuleNodes directly to the rule tree as well as edit, move and delete them. Top-down rule-driven KA can be used for instance to establish the background knowledge or ground rules in a knowledge domain or sub-domain, or to edit the existing knowledge.

With this type of knowledge exchange, a party knows some fact that it wants to tell the KBS irrespective of the number of cases seen thus far. The only limitation on the KBS's ability to receive this knowledge is its receptivity to new information, and the knowledge giver's ability to suitably codify the information for K.

[135] Thanks to (my associate academic supervisor) Lee Flax for these insights.

There is no slowing effect for top-down rule-based knowledge exchange, so if the conduit between the information giver and receiver is flawless, the expected incremental knowledge trajectory would be a straight line, with a one-to-one correspondence between the number of classes presented, and the number of classes acquired by the KBS.

7.5 Discussion and Implications

In the case-driven KA model presented, when one party first attempts to teach another party there can be a good deal of learning momentum. This varies according to the first party's will and capacity to teach, and the second party's will and capacity to learn. It will also depend on the granularity of the knowledge exchanged, e.g. what each party sees as being worthy of new classifications or otherwise. As time goes on, if the first party is not teaching anything new, and as the second party continues to learn everything that the first party has to teach them on the basis of incoming randomised repetitive cases, the velocity of learning for the second party will slow dramatically. Before that point, a party who wants to learn all that there is to learn in a given domain would be well-advised to seek alternate sources of knowledge.

As noted in section 7.2.11 (page 123), learning has a reflexive effect. The act of learning changes the learner's perceptual framework and this in turn changes how and what the learner can learn in the future. For a discussion of this feature of learning see Cooper (1998, section 3). As shown in this chapter the act of learning also changes the rate at which the learner can learn in the future. This is particularly important when there is an uneven distribution in the frequency of novel experiences.

As highlighted by Compton and Jansen (1989), many philosophers have considered the context-dependent and socially-situated nature of knowledge. We all walk in different shoes, have different experiences, and therefore have different knowledge. In the past, ESs only allowed one expert to teach them at a time – although multiple experts could sequentially add knowledge to the ESs, there was no facility for multiple experts to concurrently and collaboratively teach the KBS, and no mechanism for multiple experts to discover and resolve their classification conflicts. Since experts have different experiences with different frequencies, and different perceptions, preferences, biases, and aptitudes, it would be well-advised to draw on the rich pool of expertise offered by multiple experts in a given domain. The net result will be a summation of the case-driven and rule-driven KA trajectories offered by multiple experts. The case-driven KA model presented herein highlights the slowing

nature of case-driven knowledge exchange and calls for a more collaborative multi-expert teaching and learning paradigm.

One complexity is that often experts have different presentation needs for the knowledge. Another complexity is that often experts disagree! To solve this latter problem, what is needed is a forum in which experts can resolve their expert conflicts. As discussed previously (section 5.5, page 74), what we are beginning to realize with successes like Wikipedia (http://www.wikipedia.org) is that humans are incredibly good at knowing when they have something to offer, and when they should take a back seat. Experts are often very good at self-selecting the areas in which they can offer expertise and they are often quite willing to work through the knowledge negotiation process with other experts.

As more experts are involved in the process, more true knowledge can be accumulated, more false knowledge eliminated, more speculative knowledge confirmed or denied, the burden of populating the KBS diluted, and incoming cases can be dealt with sooner and more accurately. Where experts self-select the cases and rules that they contribute, the overall rate of learning by the KBS may be significantly accelerated.

As discussed in the next chapter, this model shows that parties can benefit from both top-down and bottom-up approaches to KA. A rule-driven KA system, in which users can add RuleNodes directly to the rule tree as well as edit, move and delete them, allows users to enter top-down general knowledge, background knowledge, or ground rules in an order and manner that makes sense to them. It allows experts to anticipate problems and share them in advance with novices; edit and correct over-generalisation or over-specialisation errors; and optimise the knowledge structure to enhance system performance. In contrast, a case-driven KA mechanism like SCRDR or MCRDR allows relative knowledge to be entered in the context of specific cases. A hybrid Case- And Rule-Driven (C.A.R.D.) approach can offer the best of both of these approaches.

7.6 Chapter Summary

This chapter has addressed the following research question:

Q5. What is the expected trajectory of the case-driven acquisition of classification knowledge?

With the relevant human factors in mind as previously described in section 7.2.2 (page 95), this chapter hypothesises that the rate of usage of tags (and hence classes) in SCRDR, MCRDR, folksomonies and tagsomonies corroborates with the case-driven KA trajectories predicted herein. Whereas rule-driven KA can be expected to follow a linear trajectory with respect to the incoming classes, case-driven KA can be expected to follow a monotonically increasing and rapidly slowing asymptotic trajectory with respect to the incoming cases[136]. If the number of classifications occurring in the domain is dynamic and increasing, an upward drift in the case-driven KA trajectory is expected.

Theorems 1 and 2 present the formulas for the expected number of classes that will be acquired as a function of the number of cases seen, and the expected number of cases seen as a function of the number of classes acquired in single classification systems. Section 7.2.5 (page 112) demonstrated how these formulas change when multiple classifications are acquired per case, and the more general Theorems 3 and 4 were presented. Section 7.2.6(page 118) demonstrated how the formulas change when the classes occur with unequal frequencies across the sample cases and Theorem 5 was presented. Finally, Section 7.2.7 (page 121) described what happens when multiple parties contribute to the knowledge base. In all cases, the case-driven KA trajectory is asymptotic such that in the expected KA trajectory 100% of knowledge is seldom if ever acquired. Mathematically, for large domains i.e. with large (m), transition to the final class in Figure 12 (page 106) is extremely unlikely since the probability of transitioning to the last class is miniscule compared to the probability that any number of multiple repeat cases will be seen.

Although the derived formulas and theorems will be useful in some contexts, the insights that the model provides to case-driven KA in general are probably of greater significance. The case-driven KA model offers important predictions for the trajectories presented in previous machine-learnt and case-based KA simulations for SCRDR and MCRDR as discussed in (Compton, Preston and Kang, 1995), (Kang, Lee, Kim, Preston and Compton, 1998), and

[136] In terms of human learning, since much of our life learning is on the basis of experience, perhaps this is why we learn so much so quickly as children, but appear to "slow down" and "become set in our ways" as we approach old age. We can shift this perception by continuously seeking new references and new inputs as life goes on. Further to this, it would be interesting to compare the case-driven KA curves with learning curves in neural networks and genetic algorithms or other biological systems and evolutionary algorithms.

(Cao and Compton, 2005 and 2006) as well as the tag-acquisition trajectories discovered by (Goldman and Huberman, 2005, p4) for folksomonies. The differences in the observed trajectories can be explained by:

- the number of possible classes (*m*) in the domain;
- the number of classes acquired per case (*c*);
- the type of frequency distribution of the represented classes;
- the number of cases available to the learning algorithm (*n*);
- changing user preference for generality or specificity in the set of classes acquired which will affect (*m*);
- any over-specialisation bias which will serve to increase (*m*);
- any over-generalisation bias which may also serve to increase (*m*). (In RDR systems this will occur if additional stopping rather than subsumption classes are acquired[137]); and
- the amount of errors in the case attributes and the classifications in the raw data that may result in extra (wrong) classes and hence a larger number of possible classes (*m*) in the domain.

This research also hypothesises that the case-driven acquisition of cornerstone cases per RuleNode in conventional MCRDR systems will follow the probabilistically expected case-driven KA trajectory derived herein (section 7.3.2, page 125).

One of the key performance criteria for KA methods is the amount of system interaction required of users. Obviously, designers of KA systems want to achieve the best exchange for the least mouse clicks, key presses and mental strain on behalf of the user. So an interesting question is: "*What is the minimum number of classes required to represent a given domain?",* and in terms of accuracy and efficiency*, "How much better or worse do different KA algorithms perform?".*

[137] Stopping RuleNodes in conventional MCRDR systems were explained on page 56.

From this research, I hypothesise that the most rapid and robust knowledge acquisition in a KBS will occur where knowledge is acquired from multiple experts who self-select their cases according to their expertise[138]; where experts are able to resolve their classification conflicts in a common forum as new cases arise; where experts have significantly different experience, perspectives, and therefore knowledge; and where the teaching-learning fit between the giver and receiver of the knowledge, and the accuracy of knowledge transfer is optimised (further discussion is provided in Appendix Q, page 473). In that case, both the rate and quantity of knowledge transfer can be expected to exceed the rate and quantity of knowledge transfer from a single expert. As well, the model of knowledge transfer presented herein calls for a hybrid case-based and rule-based approach to knowledge acquisition. A hybrid approach can be used to prepare for errors before they occur, minimise the cost of correcting errors, and enhance the overall performance of the system.

The next chapter reviews past assertions about the need for knowledge engineering (KE) in RDR systems, and advocates a collaborative hybrid case-driven and rule-driven KA approach.

[138] The different perspectives from different experts can be considered to increase the number of possible classifications (m) in the domain.

CHAPTER 8: HYBRID CASE AND RULE-DRIVEN KA

8.1 Chapter Outline

This chapter answers the following research question:

Q6. Why take a hybrid case-based and rule-based approach to Knowledge Acquisition?

In the first part of this chapter, the level of Knowledge Engineering (KE) expertise demanded by conventional RDR systems is reviewed. In the light of the KE expertise already required by conventional RDR systems, the second part of the chapter presents the reasons why a hybrid Case And Rule Driven (C.A.R.D.) approach to KA should be taken, rather than a case-only KA approach.

In the past, proponents of the SCRDR (Compton et al. 1989) and MCRDR (Kang et al. 1995) approaches have asserted that they don't need a knowledge engineer or knowledge engineering (KE) expertise (Kang, 1995 pp ii, 50) (Edwards, 1996, pp 223 – 224, 230) (Kang, Gambetta, Compton, 1996, pp 257, 261) (Richards, 1998a, p16)[139] (Beydoun, Kwok, Hoffman, 2000) (Preston, Edwards and Compton, 1994). However, both multiple and single classification RDR systems require that users:

1. Create and maintain a model of the target domain;
2. Translate cases into a form that lends itself to computer interpretation;
3. Abstract rule conditions into re-usable higher level functions;
4. Define rules;
5. Decide on rule location;
6. Optionally construct and re-use intermediate conclusions; and
7. Optionally maintain prudence profiles at each RuleNode in the system.

[139] Richards has expressed the view (personal communication, 2006) that in traditional KBS, KE involved techniques such as protocol analysis, structured interviews, card sorts, laddered grids etc to uncover the knowledge from the expert, followed by a translation of the knowledge into a model represented in logic or some other formalisation. Richards has subsequently argued that RDR allows the domain expert to become the knowledge engineer. This view has recently been supported by (Beydoun et. al, March 2005, p48).

Although the need for a knowledge engineer or analyst of some kind to set up the initial RDR domain model has been previously acknowledged (Kang, 1995, p25, p138) for dynamic target domains, the above KE activities are required on an ongoing basis. Despite the advantages of the bottom-up case-driven RDR approach, the problem of mapping an infinite and continuous stream of (probably) analogue human perceptions, interpretations and hence knowledge[140] into a discrete representation suitable for computer interpretation still remains. For example, it took 2-3 person weeks for a domain expert and a software engineer (the author) to develop just one of the functions used in the MCRDR-based support centre software trial reported in this thesis (section 12.3, page 225).

This chapter argues that if we acknowledge that KE is a fundamental requirement, even in the simplified world of RDR, it opens up the possibility of a much more efficient paradigm for knowledge acquisition (KA). Acknowledging the role and importance of top-down rule-based KA, even in a bottom-up case-driven world[141], will unlock the real power behind the RDR paradigm: the ability to acquire relative knowledge, in context[142].

8.2 Does RDR Require Knowledge Engineering Expertise?

8.2.1 Modelling the domain

SCRDR and MCRDR encourage knowledge to be maintained whilst the KBS is in routine use by allowing the user to compare the current case against one or more reference cases most relevant to the knowledge context under review, and allowing the user to differentiate the current case from the exemplar cases for known classes. This procedure has its roots in geometric triangulation[143] and the personal construct psychology introduced by Kelly (1955). Kelly's idea was that users could compare a current case with two others, determine which other case the present case is more similar to and alternatively more different from, and

[140] Further discussion of these ideas is provided in Appendix E (page 404).

[141] Please see the Glossary (page 277) for a definition of the terms top-down and bottom-up.

[142] Thanks to an anonymous PKAW 2006 reviewer who commented that in their view the main contribution of RDR is in separating knowledge engineering tasks and domain expert's tasks, rather than in reducing the knowledge engineering effort. The reviewer agreed that RDR requires more knowledge engineering than the amount indicated by past RDR literature.

[143] and hence trigonometry as applied to geographic surveying.

articulate and codify the reasons for the observed similarity and differences. Kelly's insights resulted in the repertory grid method that was further enhanced by Shaw and Gaines (1989, 1993, 2000). In MCRDR a similar comparison process is used to discover and define the features that distinguish a new case from one or more existing cases in the system.

In order for new features to be discovered and defined, new attributes and new values for those attributes need to be constantly added to the MCRDR system as it evolves. The need to create and record new attributes and values for incoming cases becomes particularly apparent when the conclusions for a current case differ from the conclusions being presented by the system, yet the case exhibits no relevant features that distinguish it from the exemplar cases for the RuleNodes being presented (Kang, 1995 pp 21, 66, 78). Edwards showed that the evolution of domain knowledge, and of the domain itself, was a prominent and sustained feature of PEIRS operation (Edwards, 1996, pp v, 105, 140, 196-197).

Defining suitable attributes and their values is a knowledge engineering function, sometimes referred to as *domain modelling*. As noted by Kang (1995, p25, p138), "*a knowledge engineer or analyst of some kind will be required in setting up the initial domain model*". However in dynamic knowledge environments domain modelling will be an ongoing KE process that is rarely complete (Kang, 1995, pp 1, 142). Kang reports that pathologists at St. Vincents quite independently decided that PEIRS would never be complete, but rather that it would keep changing as both chemical pathology and its clinical use continue to evolve (Kang, 1995, p49).

Motoda (1993, p66) highlights the difficulties associated with modelling knowledge domains: "*deciding how to partition the knowledge is a problem… which representations to use for which types of knowledge is also a fundamental and often difficult decision: the mapping from knowledge to representation is generally not clear-cut*".

As well, Shaw's study of different experts using KSS0 (1988) suggests that experts organise their knowledge of the same domain quite differently from each other, and the same expert is likely to vary his or her knowledge organization on repeat experiments. Hence the attribute-value vectors that map out the selected knowledge domain can be expected to change and mature over time in any RDR system targeted at a dynamic knowledge domain.

8.2.2 Capturing Cases

Edwards reported that the main limiting factor of PEIRS was not its technical ability to represent or reason with knowledge. Rather, most conflict related to the system's access to the same case information as was known to the pathologist (Edwards, 1996, p140). In other words, a knowledge acquisition and knowledge engineering exercise was required to codify the incoming case in such a way that the expert engine could derive useful and accurate interpretations for it.

8.2.3 Defining rule conditions and defining higher level functions

Horn (1993, section 12) notes that: "*often it is not clear what combination of discriminating attributes should be chosen to create a rule."* The difference lists provided in both SCRDR and MCRDR were aimed at simplifying this process for users, however Horn adds that: *"...as many of these attributes are experimental, they can superficially appear to be discriminating variables and make the rule base unnecessarily complex*."

In PEIRS it was found that cases could not simply be differentiated on the basis of their attributes, but rather that functions needed to be built to facilitate case differentiation. New expertise was required to correctly use the function building syntax, and to apply built functions to define suitable rule conditions (Edwards, 1996, p94). The complexity of conceiving of and building functions was seen to be one of the obstacles to comprehensive data interpretation (p144). As noted in the evaluation of PEIRS, derived values for functions did not appear in the difference lists, so valid rule conditions typically had to be created rather than simply selected (Edwards, 1996, p144). Depending on the domain, this expertise could be quite specialised. As mentioned previously, it took more than 2 person weeks for a domain expert and a software engineer (the author) to develop just one of the functions used in the software trial reported in this thesis (section 12.3, page 225).

In addition to this, Edwards (1996, p154) noted that users could be called upon to conceive of potentially complex expressions unaided, and it he perceived that some pathologists might be uncomfortable with the task. Since most pathologists have a good grounding in mathematics and statistics he felt that it was unlikely to be a major problem, and that user interface training would help users rise to the challenge.

Clearly the tasks of formulating rules by differentiating cases; and defining and reusing functions are part and parcel of engineering the knowledge to provide a conditional index between cases and the classes that represent them.

8.2.4 Locating RuleNodes

MCRDR requires that users decide on both the location and the composition of rules (Kang, 1995, pp ii, 50, 62) a task that is not always trivial (Edwards, 1996, pp 181 – 182, 194). Although the MCRDR system provides hints as to candidate locations for RuleNodes, the final decision still rests with the user (Kang, 1995, pp 20, 62). Constructing rules, and deciding on their location is a knowledge engineering task. Picking the wrong location for RuleNodes leads to over-generalisation and/or over-specialisation and hence to false positive and/or false negative errors.

8.2.5 N-RDR

As mentioned earlier in section 4.4.5 on page 62, the lack of tools for abstracting specific features to more general features was seen as an important limitation in PEIRS (Edwards, 1996, pp 120, 134, 137). N-RDR was an extension of RDR that introduced the concept of intermediate conclusions and allowed such conclusions to be re-used in rule conditions elsewhere in the decision tree (Beydoun and Hoffman, 1997). However, as noted by Drake and Beydoun (2000, p73): "*additional complications in the knowledge acquisition process (are) required to apply these approaches. These complications may add extra burden on the expert during the knowledge acquisition process*."

For example with N-RDR, users need to:

1. restrict their solution to the case at hand (Drake and Beydoun, 2000, p85)
2. decide which concepts in the hierarchy to modify (Drake and Beydoun, 2000, p87); and
3. decide whether to change the definition of concepts or reconsider the parameters passed.

At the limit of user KE capabilities, Drake and Beydoun (p87) offer the following last resort: "*we maintain that on occasions when these decisions overwhelm experts during the knowledge acquisition process, they can always revert to using propositional (as opposed to*

predicate) conditions". It is obvious from this analysis that N-RDR can demand a high level of KE skill of its users.

8.2.6 Prudence

Edwards advocated that error in KBSs should be predicted and actively managed (1996, p197) and as a result of his experience with PEIRS, he introduced the concept of prudence in RDR systems (pp i, vi, 198, 211). Feature Exception Prudence (FEP) involves managing a context profile at every RuleNode in the KBS to record the set of permissible cases at that RuleNode based on the history of cases seen by that RuleNode and validated by a human expert. The user needs to selectively filter those features that may be relevant to each RuleNode context. In the examples that Edwards provided, only numeric attributes with values falling into known ranges were subjected to FEP (p204). But for more text-based attributes, the user may need to derive their own functional measure of the similarity and dissimilarity between the values seen at different cases for the same attribute. Obviously the knowledge engineering required by Prudence strategies requires a high level of KBS understanding and skill.

Similarly, significant knowledge engineering effort would be required to tune the neural network in the Rated MCRDR system proposed by Dazeley and Kang (2004, p3) to identify cases that follow an unusual pattern of paths through the decision tree, and hence present as likely candidates for error and subsequent knowledge acquisition to occur.

8.2.7 The size and complexity of the KE and KA task is user and domain dependent

Hence although knowledge engineers may not need to be explicitly hired into the KE role in RDR implementations, because of the need to:

1. Create and maintain a model of the target domain;
2. Translate cases into a form that lends itself to computer interpretation;
3. Abstract rule conditions into re-usable higher level functions;
4. Define rules;
5. Decide on rule location;
6. Optionally construct and re-use intermediate conclusions; and
7. Optionally maintain prudence profiles at each RuleNode in the system.

KE is an ongoing requirement, even in the simplified case-driven KA world of RDR. Apart from obvious human factors, including the:

1. experience;
2. capability; and
3. capacity (including the availability)

of people assigned to the KA task, the size and complexity of the KE and KA task depends on the nature of the domain for which RDR or any other expert system[144] is being implemented, including:

1. The number of cases in the domain;
2. The number of classifications in the domain;
3. The availability of computer parse-able case data;
4. The number of attributes in the domain;
5. The dependency relationships between attributes;
6. The complexity of the ongoing case parameterisation task;
7. The nature of the structural points of differentiation available in the case data and hence the complexity of abstracting functions to represent classifications in the domain;
8. The complexity of specifying the rule conditions;
9. The amount of subsumptive overlap (and hence the dependency) of classifications in the domain;
10. The amount of polymorphic overlap of classifications in the domain (and hence the amount of exception conditions required);
11. The type and volume of other inter-class relationships that need to be represented;

[144] These issues are consistent with some of the problems experienced when applying ML and DM techniques in some problem domains as discussed in Appendix D.8 on page 403.

12. The number of different user views that need to be represented in the acquired knowledge;
13. The importance and urgency of correctness and hence of validation and verification; and
14. The changeability of the problem domain.

Finally, the characteristics of both the domain and the users will affect the propensity for KA errors to be made, the ease with which these errors can be corrected, and the importance and urgency of correcting such errors[145].

8.3 The Call for Hybrid Case And Rule Driven System

8.3.1 An over-emphasis on incremental KA and case-driven difference lists

Previous case-driven KA systems like SCRDR and MCRDR did not allow any edits to be made to the decision tree, and only allowed knowledge to be acquired on the basis of cases seen by a single expert (see also Horn, 1993, section 6). With RDR, updates to the knowledge base involve strictly incremental case-motivated additions of RuleNodes (Kang, 1995, pp 58, 70) (Dazeley and Kang, 2004). In PEIRS for instance, typographical or conceptual errors in RuleNode expressions could only be corrected with the use of "fall-through" rules (Edwards, 1996, p119, 207). As recalled by (Beydoun et. al, March 2005, p50) "*rules are never deleted or modified*"; as highlighted by (Compton, P., Preston, P. and Kang 1995, p3, in past RDR systems, "*rules are never removed or corrected, only added*"; and as assumed by (Compton, Cao and Kerr, 2004, p2), "*we cannot edit the rule*" and "*we hypothesise that there can be no advantage in carrying out these tasks in an implicit way by editing the knowledge base. Rather, there is a risk of introducing other errors in such editing".*

In his PhD thesis, Kang commented that case-based reasoning (CBR) is motivated by the idea that in conventional KBS, too much emphasis was placed on knowledge and he suggests that in CBR, too much emphasis is placed on cases (Kang, 1995, p56). In contrast, from the research reported in this thesis, it appears that conventional SCRDR and MCRDR systems have placed too much emphasis on incremental KA through cornerstone cases and case-

[145] The ease with which errors can be corrected in conventional RDR systems is discussed further in section 8.3.9 (page 153).

driven differences lists (pp 134 – 135). As noted by Kelly, Gaines, Compton, Kang, Beydoun et al., ongoing refinement by differentiating cases can be very useful. However, in the PKS experience (Appendix G.12, page 411) difference lists were rarely used since most users know what conditions they want to apply without needing to see a difference list. As well, the comparison of cornerstone cases via difference lists is only useful if the human user can cognitively make that comparison i.e. the similarities and differences between the attribute data must be able to be determined on the fly by the user. Difference lists are much more useful for structured case data rather than loosely structured case data. When the case attribute-value pairs are strongly structured, or when functions are being used to pre-process the cases and produce intelligible (human readable) attribute data, difference lists will be more human readable than when the case data is only loosely structured and functions are being applied at the RuleNodes themselves.

The use of hypothetical cases to drive KA has been common in RDR systems where essentially one begins with either a situation (a case) or a rule (a partial case) in their mind rather than a case in the real world (personal communication, Richards, 2006). Since rules are case conditions, the distinction between rule conditions and the cases that they aim to represent is sometimes not so great – the user may only collect as much of the case as is needed to define that particular RuleNode. Through the software trial of the system proposed by this research and reported in Chapter 12 (page 223), we will see that users can be very capable of dealing with a hybrid top-down rule-driven and bottom-up case-driven KA paradigm.

From these observations and experiences, it appears that humans have a richer capability to understand, abstract and express the knowledge derived from case differences than either SCRDR or MCRDR has given them credit for, and MCRDR systems have a greater need for this type of knowledge engineering expertise than perhaps previously acknowledged.

In the previous section we saw that depending on the target domain, RDR can already demand significant Knowledge Engineering skills from its users. Farquhar et al. (1995b) have previously argued that the growing popularity of object systems (languages, databases, CORBA etc) has substantially widened the group of people comfortable with working in an

object-oriented frame-based[146] and hence rule-based paradigm. As well, Cooper provides relevant insights into the real limitations of human capacity through Cognitive Load Theory (1998). In particular, he highlights the importance of not exceeding the working memory capacity of your users.

Beydoun, Kwok and Hoffman (2000) report: "*the strength of the (RDR) approach is that rules are never corrected or changed because corrections are contained in rules added on to the end*". In contrast, this research proposes that the strength of the RDR approach is its ability to acquire relative knowledge i.e. knowledge that uses an existing and recognised subsumption or exception hierarchy[147] to minimise the complexity of the elicited rule conditions; in the context of specific exemplar cases.

In the past, Compton, P., Preston, P. and Kang (1995) have asserted that: "*For an expert to build a decision tree, he or she must think about the whole domain in selecting an attribute to go at the top of the tree. With RDR the expert is asked simply to justify their conclusion in context in the same way as in normal human discourse*". In contrast, this research indicates a middle ground in which some experts are very comfortable working in both a hybrid rule-driven and case-driven paradigm. By separately tracking the live versus registered case-RuleNode associations, and hence allowing users to delay the resolution of knowledge inconsistencies created by case drop-down scenarios (discussed previously in section 6.5, page 84), experts at most need to consider the context of a given RuleNode when editing it, as opposed to the entire problem domain.

8.3.2 Solving the MCRDR Repetition problem

Although MCRDR made significant headway in solving the repetition problem that occurred when SCRDR was applied to multiple classification problem domains, it too suffered from problems of repetition where identical classifications were served up from multiple different locations in the decision tree[148]. Apart from identical pathways being repeated in different

[146] Minsky introduced the concept of frame-based systems (Minsky, 1975).

[147] The use of RDR in acquiring both subsumption and exception hierarchies was discussed in section 4.3.2.4 (page 60).

[148] The MCRDR repetition problem is also discussed in sections 8.3.10 (page 155) and 13.4 (page 245).

orders, identical classifications could be arrived as a result of different heuristics (personal communication, Richards, 2007).

Edwards showed that as the MCRDR PEIRS knowledge base grew, the mean MCRDR rule trace generated per case also grew (pp 190-191), indicating that the average replication in the MCRDR worsened in proportion to the number of cases seen by the system, and hence in concert with the amount of knowledge acquired. After 262 cases, the MCRDR PEIRS system contained 22 conclusions and 77% of them appeared in more than one RuleNode in the knowledge base (p191). Further to this, in Kang's machine-learnt studies (Kang, 1995, p80), no cost was considered as regards the repetition of conclusions i.e. interpretations across the system.

When MCRDR is translated to a problem domain in which users need their cases evaluated in real-time, the negative impact on the responsiveness of the system could be significant. As well, since the KA of interpretations was the most time-consuming KA task of all in PEIRS (Kang, 1995, p40), repetition can be viewed as a significant KA and maintenance problem in conventional MCRDR systems.

The problem of identical classifications being arrived at through different heuristics is particularly likely to arise in conventional MCRDR systems that provide rules with only conjunctions (ANDs) rather than ORs (Kang, 1995, p66) since alternate pathways to the same conclusion can only be represented by physically separate pathways[149].

PKS went on to develop comprehensive post-processing strategies to deal with the MCRDR repetition problem (Appendix G.12, page 411), and Beydoun, Kwok and Hoffman (2000) derived incremental KA strategies to solve the generalised RDR repetition problem by promoting order independence in the purely random case-driven KA RDR world. Rather than applying these extra knowledge engineering strategies, an alternative that could result in more rapid KA would be to allow users to perform top-down rule-driven KA. Users would be able to add, edit and delete attributes, cases and RuleNodes (including their classification labels and their conclusions); and move RuleNodes. This would allow users to eliminate redundant RuleNodes, and expand the rule conditions at RuleNodes to make them more accurately

[149] In contrast, the use of ORs in the rule conditions, and the use of the shared child RuleNode structure (section 13.4page 245) in the system proposed by this research, will alleviate this problem.

represent their corresponding classifications. Users could perform these additional tasks as best fits the situation while being informed as desired by the current case context, by the relationship between existing RuleNodes, and by reference cases currently indexed by the system[150].

Drake and Beydoun (2000, p83) suggest that: "*the key strength of the RDR knowledge acquisition framework is that the resultant knowledge base can be easily modified. This has reasons: firstly the cause of failure of an RDR knowledge base is automatically determined. Secondly, newly added rules do not impact past seen cases*." While this may have been true for NRDR implementations in which case drop-downs were not permitted, as discussed previously (section 6.5, page 84), newly added rules did impact past seen cases even in conventional SCRDR and MCRDR systems. It's just that the impact was never tracked. In the system proposed by this research, the validation and verification of knowledge is continuously supported by a separation between the *live* and *registered* case-RuleNode associations that enables the impact of knowledge evolution on past cases of interest to be tracked[151], and (optionally) after a passage of time, the relevant knowledge corrected.

8.3.3 Lessons from PKS

In our interview with Lindsay Peters, the Chief Technology Officer of Pacific Knowledge Systems (PKS) we learnt that in practice a dual case-based and rule-based approach was needed for users working with the MCRDR knowledge base. In that interview, the following specific comments were made (see Appendix G.12 on page 411):

1. The knowledge base needs to have a balance between being unstructured and structured.
2. Incorrect rules are never changed since "rules are cheap". However PKS is looking at relaxing those conditions.

[150] Allowing users to combine RuleNodes with shared child RuleNodes will also help solve the MCRDR repetition problem. This approach is presented later on in section 13.4 (page 245).

[151] Case tracking was introduced in section 6.5 (page 84). Further information about case tracking is provided in Appendix O.4 (page 454).

3. Difference lists are rarely used since most users know what conditions they want to apply without needing to see a difference list.

The first point implies that a degree of top-down rule-based knowledge engineering is required in order to maintain a suitable structure in the KBS. It also implies that some top-down rule-driven KA did occur.

The second point indicates that at that time PKS was considering making RuleNodes editable.

As mentioned earlier, the third point indicates that users are often comfortable with entering rules using top-of-the-head knowledge, rather than solely relying on bottom-up knowledge driven purely by the differences between cases.

8.3.4 Reducing the scope and span of RuleNodes in order to ease the codification task

Edwards (1996, p128) quotes Jackson (1990) in suggesting that in conventional expert systems, knowledge engineers could only add between two and five rules per day. Edwards and others have previously conjectured that as the traditional knowledge bases grow, the knowledge acquisition bottleneck worsens because of the inherent difficulty knowledge engineers have in dealing with such systems. According to Richards (personal communication, 2006), conventional expert systems attempted to capture global knowledge that is valid in all situations. Hence in conventional expert systems it could take people hours to work out what changes would be needed across the knowledge base to ensure the new rule being added didn't make the existing knowledge inconsistent. A complex network of pathways, involving a combination of intermediate and final RuleNodes might be used, so it wasn't always clear how changing one node might affect other pathways. Users might have had 10 rules to add, but to ensure no errors crept in they might need to spend 2-5 days. To address this problem approaches such as the use of Truth Maintenance Systems were developed to identify and resolve inconsistencies.

In retrospect, it seems that the KA bottleneck resulted from complicated coupling and dependencies between RuleNodes and cases. RuleNodes were used to perfectly represent their classifications, without redundancy or duplication in the decision tree[152]. The

[152] For example, the set (A_1, A_2, A_3) could have been represented by a single RuleNode (A) in conventional expert systems, but by three separate RuleNodes in RDR.

complexity of the rule conditions at each RuleNode and the number of cases affected by changes to each RuleNode would have been much greater in conventional KBSs where users aimed at zero redundancy, and hence more "perfect" RuleNodes with a much wider scope and span[153]. Since the impact on all affected cases was to be dealt with at the time that the RuleNode was added rather than at some more convenient time in the future, the task of adding each new RuleNode could have been very laborious and the complexity of that task could be prohibitive[154].

Compton and Jansen have previously claimed that the major feature of ripple down rules is that they can be added to a knowledge base far faster than conventional rules since the rules are added without modification, and only in a local context[155] (1989, p12). They claim that "*it is not too difficult to add 10 rules per hour in contrast to the often mentioned industry figure of 2 rules per day*". RDR allows knowledge to be acquired in an ad-hoc fashion. Redundancy and duplication of classifications and hence conclusions in the decision tree is permitted and frequent, even in MCRDR systems. The acquisition of rule conditions relative to ancestor RuleNodes means that the rule conditions at RuleNodes can be simplified and each RuleNode can affect only a handful of cases compared with the more "perfect" RuleNodes referred to in the Jackson (1990) example.

This thesis advocates that the real benefit of RDR is the ability to acquire relative knowledge, in context. There is no evidence that it's any faster working from a top-down or bottom-up perspective per se. The process of constructing the rule and its interpretation is the same. It's just that it is easier to see the need for and conceive of rules when the KA is grounded in cases. As well, it is easier to construct rules within a subsumption or exception hierarchy in which the ancestor rule path can be easily inherited, and the scope and span of each RuleNode is significantly reduced. Apart from the presence of a relative case and RuleNode context, the *actual codification task* of the RuleNode conditions is identical in both conventional rule-

[153] *Scope* refers to the set of cases classified by the RuleNode (see Beydoun et. al, 2005, p52), where-as *span* refers to the set of cases dependent on the RuleNode and its child RuleNodes. In (Beydoun, Kwok and Hoffman, 2000, p4) *span* is defined as the *domain*.

[154] As mentioned previously, Cooper highlights the importance of not exceeding the working memory capacity of your users through Cognitive Load Theory (1998).

[155] In RDR, the context is local relative to a possibly much wider subsumption or exception hierarchy.

based expert systems, and RDR case-based expert systems. The main difference is the scope and span of each RuleNode, and hence the complexity of the rule conditions at the RuleNode and the number of cases affected by it.

8.3.5 The rate of KA declines naturally in case-driven KA systems

In Chapter 7 a predictive stochastic model for case-driven KA was derived that showed that it is very unlikely that a system that only uses case-driven KA in a large problem domain[156] will ever achieve complete knowledge. The mathematical model calls for a hybrid case-driven and rule-driven KA system. As discussed previously (sections 7.5 on page 128, and 7.6 on page 129), a hybrid approach can be used to prepare for errors before they occur, minimise the cost of correcting errors, and enhance the overall performance of the system.

The analysis in section 7.2.6 (page 118) showed that the rate of accumulation of RuleNodes and hence the rate of improvement in system accuracy is impacted by the KA properties of random and unevenly represented classes. For domains with a large number of classes (m), the case-driven KA trajectory is asymptotic such that in the expected KA trajectory 100% knowledge is seldom if ever acquired. Mathematically, for large domains, acquisition of the final class is extremely unlikely since the probability of seeing an exemplar case for that novel class is miniscule compared to the probability that multiple exemplar cases for previously seen classes will present themselves. The variance about the expected number of cases required to achieve the last few nuggets of knowledge is enormous.

The expert maintaining PEIRS was reported to take 2-3 minutes on average to add a new rule in the SCRDR system (Kang, 1995, p129) (Edwards, 1996, p193). Once the need for a rule was perceived, the major problem for staff was not in creating the rules, but in deciding on the classification, its wording, and the most helpful form of interpretation to provide (Kang, 1995, pp 40, 129). Kang reports that the conditions to be selected were always obvious, since they were the reason the expert recognised that the conclusion was wrong. But after 25 months of operation, only 3.3 rules an average were added to PEIRS each day (Edwards, 1996, p97). Hence the type of software strategy employed to add rules to the knowledge base is not the only factor that contributes to the rate of KA.

[156] i.e. a domain with a large number of classes to be acquired (m).

The rate at which new rules can be added to the KBS also depends on *users seeing the need for rules in the first place*. Obviously, as a KBS matures to more closely represent the chosen problem domain, the frequency with which new rules need to be created will decline. As a further example, after 29 months of operation of the LabWizard MCRDR software at PKS, the rate of rule addition across the 19 different sub-domains subsided from a peak of 1 hour per day to less than 20 minutes per day. In 2005 generally it was taking a minute or two to add a new rule implying that only 10 to 20 rules per day were being added to the 16,000 RuleNode system (Compton et al. 2005).

As noted by Horn (1993, see his abstract), "*the manual acquisition of rules using monotonic maintenance methods, such as ripple down rules, can be laborious*". In contrast, and as discussed in section 7.5 (page 128), a hybrid Case And Rule Driven (C.A.R.D.) approach allows users to enter top-down general knowledge, background knowledge, or ground rules in an order and manner that makes sense to them. It allows experts to anticipate problems and share them in advance with novices, as well as edit and correct over-generalisation or over-specialisation errors, and optimise the knowledge structure to enhance system performance. The manner in which the system proposed by this research supports a C.A.R.D. approach is described later on in Chapters 9 and 11 (for an example of the Rule Tree view in which one can edit and move RuleNodes, see Figure 40 on page 219). The reasoning mechanism that allow this to occur and that protects the knowledge system against an inconsistencies is described later in section 11.7 (page 210).

For example, Horn (1993) notes that: "*It is sometimes possible that a more simple and general rule can replace a number of overly specific rules. These simpler rules, not only reduce the size of the knowledge base, but can improve the accuracy of the system and improve comprehension*." The KA model derived in Chapter 7, indicates that such fundamental improvements to the KBS will be most easily and rapidly acquired in a top-down rule-driven manner.

8.3.6 Ground Rules versus Experience based knowledge

In their evaluation of simulated expertise, Compton, P., Preston, P. and Kang (1995) report their belief that "*eventually all knowledge acquisition reduces to asking an expert, at least implicitly, to deal with cases*". This seems to be a reasonable conclusion for the end-game acquisition of knowledge in a given domain, but this research has found that it doesn't reflect

the reality during the start-game. For example, in PEIRS the first 198 rules were added off-line in a top-down manner while interfacing problems were sorted out (Kang, 1995, p 34-35) (Edwards, 1996, p89). Similarly, in the support centre MCRDR trial reported by this research (section 12.9.4 on page 236), the first 21 RuleNodes were most expediently added in a top-down rule-driven manner. As well, as noted in section 8.3.3 (page 144) it was found that PKS needed to structure their MCRDR knowledge base in a top-down manner. Therefore while cases are necessary to test a system, this thesis argues that RuleNodes are useful on their own in characterising a system. To improve the human-computer interaction experience, users should therefore be able to add RuleNodes in a top-down manner. The reciprocacy between the case and RuleNode views discussed later on in section 13.2 (page 242), and the "look-ahead" feature proposed in section 12.12 (page 239) can be used to inform and warn users of the likely impact (if any) of their proposed top-down RuleNode changes.

In their study of crisis management ontologies to assist with scenario analysis and decision making in the military, Cohen et al. (1999) showed that the availability of prior knowledge i.e. background knowledge or ground rules accelerated the construction of KBSs. They showed that prior knowledge in imported ontologies accounted for a roughly constant rate (60-66%) of reuse, irrespective of who had developed the imported ontologies. They noted that the real advantage of these ontologies comes from helping knowledge engineers organise their knowledge bases along sound ontological lines. The researchers noted that one of the additional benefits of background knowledge in ontologies is the recognition and creation of subclasses that inherit and augment the axiomatic behaviour of their parent classes.

When starting to learn in any domain, the age-old concept of *ground rules* is relevant where ***ground rules*** are for example, "*a common set of agreed standards in some process… that allow meaningful dialogue to proceed with the aim of minimizing conflict"*[157]. Ground rules can comprise *the codified rules of the game*. The design implication is that in order to capture both the case-independent ground-rules, and the case-dependent refinement rules, the system needs to handle user interaction in both a top-down rule-driven and a bottom-up case-driven manner.

[157] http://en.wikipedia.org/wiki/Ground_rules (Jan 2006)

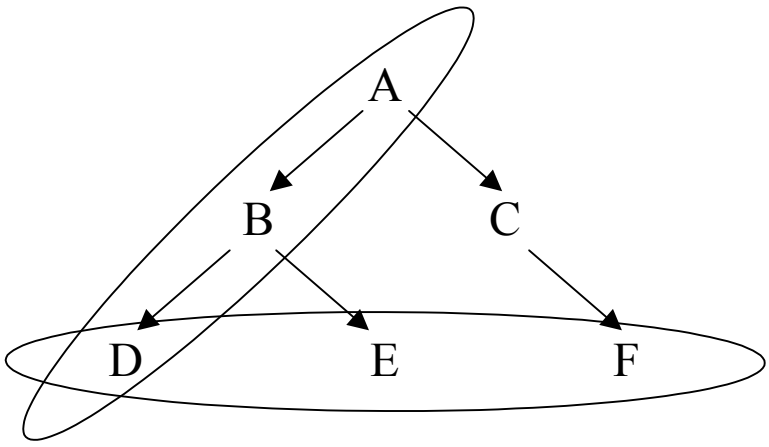
8.3.7 Lessons from DMQL

The Data Mining Query Language (DMQL) offers some further insights to case-based versus rule-based knowledge acquisition. DMQL has been proposed to augment traditional SQL functions with those found in OLAP (online analytic processing) and data mining applications (Dunham, 2003, p18). DMQL allows access to background knowledge for data mining, such as concept hierarchies. The heart of the DMQL statement is the rule specification portion, which notably contains the following different three types of rules:

- characteristic rules – these include the conditions satisfied by almost all the records in a target class
- discriminate rules – these include the conditions that differentiate the target class from other classes
- classification rules – these are the conditions that are used to define a particular classification for the data.

For example, in the MCRDR rule tree in Figure 17, the rule A (at RuleNode A) is an example of a characteristic rule because in includes the conditions satisfied by all the other RuleNodes and hence records classified by the decision tree. The rule D (at RuleNode D) is an example of a discriminate rule because it includes the conditions that differentiate the target class from other classes. An example of a classification rule is (A and B and D) which is comprised of the following characteristic rules (A and B) and discriminate rule (D).

Figure 17: Characteristic versus Discriminate versus Classification Rules



Both characteristic and discriminate rules can be acquired via top-down or bottom-up KA methods. However, there may be a case for arguing that purely characteristic rules are more easily acquired via top-down rule-driven KA methods, whereas purely discriminate rules are more easily acquired via bottom-up case-driven KA methods. For rules that are both characteristic and discriminate, both KA approaches can work.

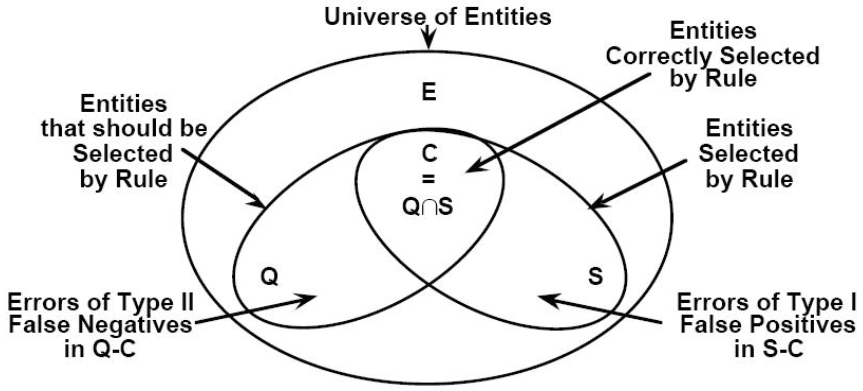
8.3.8 The Nature of Classification Errors

Compton (2000) shows that errors occur in RDR systems. He provides an analysis of the types of errors that experts make in building KBSs and together with (Cao and Compton, 2005, 2006) expert error is reduced to that of:

1. Over-generalisation resulting in false positives i.e. some cases are inappropriately included in the classifications. Over-generalisation can be quantified in terms of sensitivity. The intuitive response to an over-generalisation is to add rule conditions.
2. Over-specialisation resulting in false negatives i.e. some cases are inappropriately excluded from the classifications. Over-specialisation can be quantified in terms of specificity. The intuitive response to an over-specialisation is to remove rule conditions.

Gaines has previously highlighted the problem of false positives and false negatives in his presentation of the *Induct* machine-learning algorithm (1989, p2). Figure 18 reproduces his universe of entities **E**, with the target predicate **Q** and set of possible test predicates **S**.

Figure 18: Problem of empirical induction



Reproduced from Gaines (1989, p1)
with the kind permission of Brian Gaines.

Previous SCRDR and MCRDR implementations do not allow RuleNodes to be edited (Kang, Gambetta, Compton, 1996, p260) and therefore changed so if an over-generalisation error occurred towards the top of the rule tree, the entire sub-tree could require patching with exception rules, which could be a very costly knowledge acquisition exercise. In contrast, if an over-specialisation error had occurred towards the top of the rule tree, the entire sub-tree might require duplication, which again could be a very costly knowledge acquisition exercise.

In conventional SCRDR and MCRDR, it was thought that the integrity of the knowledge base would be maintained by constraining the expert to create valid rules (Kang, 1995, p24, 55). For example Beydoun, Kwok and Hoffman (2000) offer the following viewpoint about the knowledge elicited by the RDR case-driven paradigm: "*Corrections entered by the expert are always guaranteed to be valid, because of the way conditions of new rules are chosen*". However this doesn't take account of the fact that if one is to avoid over- and under-generalisation errors, the classification and hence conclusions provided at any given RuleNode must *exactly match* the rule conditions represented by that RuleNode.

For example, in a conventional SCRDR or MCRDR system one can validly say in the decision tree: "if it swims, it's a fish". But this is obviously an over-generalisation since most mammals also swim, and they are not fish. It could be a very onerous task for a user to provide exceptions to the fish RuleNode for every incoming subclass of swimming mammal seen by the system, not to mention pointless if the user recognised their error and they were willing and able to edit the fish RuleNode to more accurately specify its rule condition.

In conventional MCRDR, a strategy that tends to add rules at the top of the rule tree will cover the domain more rapidly, but with a greater likelihood of over-generalisation errors (Kang, 1995, p63). In contrast, a strategy that tends to add rules at the bottom of the rule tree will cover the domain more slowly, and with a greater likelihood of over-specialisation errors. Kang (1995) suggested the two alternate strategies of asking a user to either (i) select the important data used in reaching a conclusion, or else (ii) delete the irrelevant data. If the selected data satisfies an existing pathway in the decision tree, the system places the new RuleNode at the location that the selected conditions reach down to[158].

[158] Kang (1995, p64) notes that the performance of his proposed approach to incremental RuleNode acquisition in MCRDR was not fully evaluated.

Rather than focussing entirely on the rule conditions and placement of RuleNodes in the decision tree, a greater focus on the nature of the classifications relative to their condition paths may improve the effectiveness of KA in MCRDR systems. Clearly whether a RuleNode is too general or too specific depends not just on its rule conditions, but on the classification it aims to represent by those rule conditions. In the ideal scenario, the set of conditions given by the rule path for each RuleNode, and the classifications given at the RuleNode, will be increasingly aligned to each-other by experts refining any or all of the knowledge (including either the rule conditions, the RuleNode location, or the classifications and conclusions) as the knowledge base evolves. As we will see later (section 12.2, page 223), in the support centre software trial conducted by this research, when given the opportunity, users relied heavily on facilities to edit and change RuleNodes, including their rule conditions, classifications and conclusions, in order to improve the knowledge represented by the expert system.

8.3.9 The Cost of Correcting Errors

In the SCRDR simulations developed by Compton (2000), the relative cost of errors was not developed and a simplifying assumption was made that all errors are equal (Compton, 2000, p5 paragraph 3). The same assumption was made for the simulations reported in (Cao and Compton, 2005, p3 paragraph 3) for SCRDR, Flat RDR[159] and KA via the Composite Rules framework[160]. No formal study has been made of the relative costs of overgeneralisation and overspecialisation errors made at different levels in a multi-level MCRDR decision tree. However, it is conceptually clear that the cost of correcting errors using only local patching is greater for rules at the top of a multi-level MCRDR decision tree, than for errors lower down in the decision tree, purely because of the volume of cases incorrectly classified as a result of these errors in the case of over-generalisation, and because of the volume of sub-tree RuleNodes and hence Cases inappropriately excluded by these errors in the case of over-specialisation. It can be inferred from the results of (Cao and Compton, 2006) that adding a taxonomic structure in the decision tree will result in more rapid domain coverage for a

[159] Flat RDR is an implementation using an MCRDR decision tree with depth < 2.

[160] Note that the simulations offered in (Cao and Compton, 2006) relied on a subset of the simulation framework described in (Cao and Compton, 2005) where Flat RDR was applied to a single classification problem domain.

simulated expert with overspecialisation and overgeneralisation error rates of up to 30%. Hence as previously discussed in section 8.3.6 (on page 148), allowing a multi-level taxonomic structure as provided by conventional multi-level MCRDR is shown to be an important component of KA systems.

Compton notes that the type of error that is more acceptable will vary with the domain (Compton, 2000, p6 paragraph 2). Those simulations (Compton, 2000) emphasised the impact of over-generalisations which easily lend themselves to augmentation further down in the decision tree, and they discarded the impact of over-specialisation errors, which obviously can be very costly in terms of knowledge acquisition for well-developed sub-trees (Compton, 2000, p12 paragraph 2). This position was possibly taken since, as discussed previously in section 4.3.1.3 on page 49, RDR users tend to differentiate cases using the least number of rule conditions, resulting in an over-generalisation bias in the knowledge acquired.

The simulation study by (Cao and Compton, 2005, p7 paragraph 2) found that the occurrence of overgeneralisation errors created more error overall in SCRDR, Flat RDR, and Composite Rules knowledge bases, than the occurrence of overspecialisation errors. In the case of Flat RDR and Composite RDR there was no depth in the resultant rule trees, so the cost impact of an overspecialisation error is far less than in an MCRDR implementation where an overspecialisation error can occur at a RuleNode with many dependent RuleNodes, for example towards the top of a multi-level decision tree.

The simulation study by (Cao and Compton, 2006) again used Flat RDR (as opposed to multi-level conventional MCRDR) and studied the impact of a taxonomic hierarchy and the use of cornerstone cases in the presence of both overspecialisation and overgeneralisation errors.

Over-specialisation errors are quite likely to occur in dynamic knowledge environments such as the ICT domain, where several if not many of the incumbent rules in the decision tree refer specifically to current systems e.g. software or hardware versions in order to differentiate these systems from their legacy counterparts. When future systems comprising e.g. future software or hardware releases are deployed, existing RuleNodes which perhaps should still fire may not, for example when the software or hardware versions identified at existing RuleNodes have not anticipated the new release.

In the ICT trouble-shooting domain, both types of errors can have a significant impact. The false positive over-generalisation errors can lead to errors of seismic proportion if the wrong

solution is implemented for a given case, for example causing loss of data, legal liability, and/or undue system downtime at a customer site. Similarly, the false negative over-specialisation errors will mean that the trouble-shooter does not find the appropriate solution and an excessively long time-to-resolution for the customer may result.

8.3.10 Performance implications of System Compactness and Balance

In their use of simulated experts in evaluating knowledge based systems using subsets of the Chess, TicTacToe, and Garvan datasets in the Irvine data repository, Compton, P., Preston, P. and Kang (1995) found that for the smartest expert, selecting 4 conditions gave the entire intersection of a classification in nearly all cases. As well, they found that on average, only 2-3 rules needed to be satisfied by a case in order to reach the correct solution. The researchers concluded that their results did not indicate "*any pressing need to reorganise the KB*", and provided that "*a conclusion from this work is that the effort that is put into trying to organise knowledge into an optimal model is often unnecessary*". However, the Chess and TicTacToe domains were both bi-conclusion domains and the strategy employed to add RuleNodes to the KBS favoured RuleNodes being added at the topmost level (Kang, 1995, p84). Hence a flat rule tree was possibly generated with a typical depth of 2 RuleNodes, with the first level covering all the instances where one of the two possible conclusions would be achieved with stopping nodes at the second level to manage the exceptions. It seems likely that bi-conclusion domains like Chess and TicTacToe constructed in this manner are unlikely to generate any significant depth in the acquired rule tree.

In contrast, Kang reports that at a point where the SCRDR-based PEIRS system had over 2110 rules, rule paths had an average of 4-5 RuleNodes per pathway, and a maximum of 19 RuleNodes, where each RuleNode had an average 1-2 conditions with a maximum of 6 conditions (Kang, 1995, p36). Rules were restricted to conjunctions of conditions. This reveals significant depth in the condition paths of the acquired knowledge base. Note however that this would have been exacerbated by the problem of trying to represent a multiple classification problem domain (arterial blood gases in pathology testing) using a single classification RDR framework.

As can be reasonably expected[161], the datasets used by the Compton, P., Preston, P. and Kang (1995) simulations included only a subset of all possible cases, and resulted in a finite subset of RuleNodes with limited domain coverage. For example, the Chess KA simulations resulted in a worst-case ("dumb" expert) maximum of 90 RuleNodes; the TicTacToe simulation resulted in less than 200 RuleNodes; and finally the Garvan simulation resulted in less than 1300 RuleNodes. Less RuleNodes were achieved for the simulated experts with higher competence. As well, 60% of the acquired RuleNodes were stopping RuleNodes for all 4 trials involving the Garvan thyroid domain, and 40-60% of RuleNodes were stopping RuleNodes for the chess and TicTacToe domain (1995, Kang, pp 117-119).

As discussed previously in section 2.2 (page 14), and as we shall see later in section 10.1 (page 169), the support centre context targeted by this research required that up to 5,000 cases per day be handled, and that multiple permutations of more than 10,000 different hexadecimal error codes and resultant classifications be responded to. Past assertions about the unimportance of balance and compactness of the decision tree appear therefore to have been made in the context of experiments and simulations that used a relatively small selection of datasets. New challenges arise when faced with the performance demands in the support centre context.

As well, and as discussed previously (section 8.3.2, page 142), although MCRDR significantly reduced the repetition problems experienced when SCRDR was used in multiple classification domains, repetition was still a significant problem, even in conventional MCRDR systems.

In Kang's machine learning evaluation (Kang, 1995) there was no systematic KA strategy to seek out a subsumption hierarchy for the acquired classifications. Instead, for the modelled novice and moderate experts a systematic over-generalisation bias was introduced by placing all new RuleNodes at the top of the rule tree (Kang, 1995, p84). While it has been shown that MCRDR is better than SCRDR at ensuring that under-generalisation and hence the repetition of classifications and conclusions does not occur (Kang, 1995, p104), MCRDR systems that fail to take advantage of an accurate subsumption hierarchy afforded by taxonomic domains

[161] Note that for infinite domains, cases can only ever be a brittle representation of reality. The presented cases will only ever be a subset of the cases that can occur in reality.

mean that interpretations may not be fully referenced and reused in the system. Depending on the domain, this can lead to a much heavier maintenance load for users.

Compactness and balance become critically important where the cases are to be evaluated against the decision tree in real time, for example as in an interactive and recursive implementation. After his experience with PEIRS, Edwards (1996, pp 138, 185, 194) concluded: "*response time will be an important issue for real-time interactive systems*", and "*top level rules need to evaluate all cornerstone cases in the system. This can slow the maintenance task considerably. The extent of this problem with very large knowledge bases needs to be explored."*. Hence, the size of the decision tree and the number of cases that it must handle will determine the significance of the properties of compactness and balance.A system providing multi-user real-time evaluation of cases against the knowledge base cannot afford to keep users waiting. According to one human computer interaction (HCI) study, a tolerable user wait time is about 2 seconds (Nah, 2004). Users will wait somewhat longer if there is feedback that something is happening. Given the delays inherent in network based communication, albeit on a private network or via the public Internet, it seems intuitive that a system with in excess of 1000 RuleNodes might need some degree of decision tree subsumption and / or balancing in order to maintain transaction integrity whilst providing for general system availability, and providing a response time to networked users in the order of 2 seconds.

According to the PKS experience (Appendix G.7, page 410), conventional MCRDR knowledge bases with thousands of rules can process 50 cases per second i.e. 0.02 seconds per case. Therefore for n users, each with 1 case being concurrently processed by the same system in real-time, this translates to a waiting time of n * 0.02 seconds. If you have 100 concurrent users evaluating cases using a conventional RDR approach, your wait time would already be more than 2 seconds.

In addition to this, the PKS experience offered that when making changes to the knowledge base, 4 transactions per second were possible. So for every concurrent case or RuleNode write, concurrent users of a hypothetical reasonably sized conventional RDR system could expect a further 0.25 second delay. Recall the need for users to work-up their cases, and the dynamic nature of the knowledge being captured. Finally, consider the inherent delays associated with a globally accessible real-time client-server solution that needs to share an

already busy corporate network. (In contrast, PKS's LabWizard product processes the vast majority of its cases off-line in batch mode.) It would seem that because of the added delays involved new strategies are needed to deal with multiple users concurrently evaluating cases against and writing to an RDR system in real-time.

In the system proposed by this research, a preliminary population of the decision tree with all of the available equipment error codes gave rise to in excess of 10,000 RuleNodes. Given that the system is required to handle up to 5000 cases per day, and be viewed by at least hundreds of concurrent users, and possibly thousands, decision tree compactness and balance are obviously important factors. If top-down rule-driven KA can more rapidly solve problems of redundancy and imbalance, then it makes sense to embrace and combine it with bottom-up case-driven KA.

8.4 Chapter Summary

In answer to the research question:

Q6. Why take a hybrid case-based and rule-based approach to Knowledge Acquisition?

In the first half of this chapter, it was shown that both multiple and single classification RDR systems can require significant knowledge engineering efforts depending on the nature of the problem domain, for example to dynamically create and maintain a model of the target domain; translate cases into a form that lends itself to computer interpretation; define rules and abstract conditions into re-usable higher level functions; decide on rule location; optionally construct and re-use intermediate conclusions; and optionally maintain prudence profiles at each RuleNode in the system. The characteristics of the domain, and the users will determine the ease with which knowledge can be acquired, even for RDR based expert systems.

In the second half of this chapter, it was argued that a hybrid case-based and rule-based approach is justified when users possess characteristic and / or background knowledge that is ready to share in a top-down rule-based form. This top-down rule-based knowledge can be used to prepare for errors before they occur, minimise the cost of correcting errors, and enhance the responsiveness and completeness of the KBS. Previously (Compton, Cao and Kerr, 2004, p2) have asserted that: "*the only possible way of characterising, testing or evaluating a knowledge base is via a set of cases*"; that "*the testing is only as good as the case*

available"; and that "*regardless of the quality of cases, they are the only way of characterising the system*"[162]. In contrast, this thesis argues that while cases are necessary to test a system, RuleNodes are sufficient on their own to characterise a system, even those that have been added in a top-down manner without any specific reference to a case. This research concurs with the arguments put forward by (Beydoun et. al, 2005, p48) and previously described in section 5.2 (page 66) that a state of consensus between experts reflects a mature model. Consensus can be built with specific and current cases in mind, or with future anticipated cases in mind. It can also be built with general classifications and rules in mind. In other words, consensus can be reached regardless of whether the model is built in a top-down, bottom-up, or hybrid manner.

Whereas conventional rule-based expert systems may have given primacy to the *classification* (section 3.5.1, page 36), and RDR-based expert systems gave primacy to the context-dependent *case*, the 7Cs approach proposed by this research gives primacy to the context-dependent *consensus* between the multiple agents or users who use the system[163]. The insight is that the *context* of knowledge depends not only on cases, but also on the experts who use that knowledge.

Consensus can be achieved by several different conflict resolution mechanisms, for example voting or negotiating[164]. In the proposed 7Cs system negotiation is supported by separating the public (*live)* and private (*registered*) views of KA participants as described previously in section 6.5 (page 84) and encouraging users to resolve any inconsistencies that crop up between their private view of the knowledge, and the shared public view of the knowledge. Voting is supported by maintaining a count of the number of cases *registered* by users for a given RuleNode. A comparison between the voting and negotiation strategies is left for future researchers, but the end result of consensus and hence validation and verification is

[162] Later on in (Compton, Cao and Kerr, 2004, 9) the authors note that they are not yet able to prove the conjecture that all knowledge acquisition can be reduced to correcting and adding to the knowledge, using only the two relations of sequence and correction.

[163] Shaw and Gaines (1989) identified consensus as one of several outcomes of conflict as previously shown in Figure 6 (page 68).

[164] Many thanks to an anonymous Computer Supported Cooperative Work in Design (CSCWD) reviewer (2007) for the comments about voting versus negotiating in building consensus.

supported through the notion of *approved* RuleNodes as described later in Appendix O.5 (page 456).

It was beyond the scope of this thesis to perform user trials in a variety of different problem domains to quantitatively determine how each of the dimensions outlined in section 8.2.7 138) together with a top-down or bottom-up KA paradigm really affects the size and complexity of the KE and KA task and therefore the usability and functional effectiveness of the resultant expert system. This task is left for future researchers to study in much greater depth.

This chapter has argued that acknowledging the role and importance of top-down rule-based KA, even in a bottom-up case-driven world, can unlock the real power behind the RDR paradigm: the ability to acquire relative knowledge, in context. The system proposed by this research allows for the creation, editing, and deletion of attributes, cases, and RuleNodes; as well as the relocation of RuleNodes in the decision tree. Significantly, the recording and maintenance of the *live* and previously *registered* case-RuleNode associations allows the impact of both top-down rule-based and bottom-up case-based KA to be identified and reported, and it allows knowledge inconsistencies to be dealt at the user's discretion and convenience.

The next chapter presents the top-level design of the proposed 7Cs system.

CHAPTER 9: 7CS TOP LEVEL DESIGN

9.1 Chapter Outline

Recalling from section 1.3 on page 3 that:

> *a further sub-goal (of this research) was to develop an approach that would be flexible enough to support collaborative trouble-shooting and classification in other domains such as botany, zoology, biology, chemistry, pathology, geology, and financial markets;*

this chapter presents top-level conceptual representations of the proposed system. A novel 7Cs model is offered that supports the Collaborative Classification and Configuration of a stream of incoming Cases via a relational structure of ConditionNodes, Classes and Conclusions (hence 7Cs).

The 7Cs model stretches the MCRDR algorithm to encompass collaborative classification and offers a very lightweight information broker that can act as an index to knowledge resources across an organisation's Intranet and / or across the broader Internet. A key feature provided by the 7Cs model in support of collaboration is the separation between the *live* case-RuleNode associations that represent the current global truth calculated and recorded by the 7Cs expert system (i.e. the knowledge base), and the *registered* case-RuleNode associations that represent the current truths of individual contributors to the expert system.

The 7Cs model has been demonstrated and tested as part of this research via a prototype system known as *FastFIX*. The detailed design for the FastFIX prototype is presented later in Chapter 11 on page 176, and its web-based application shell is described in Appendix M (page 424).

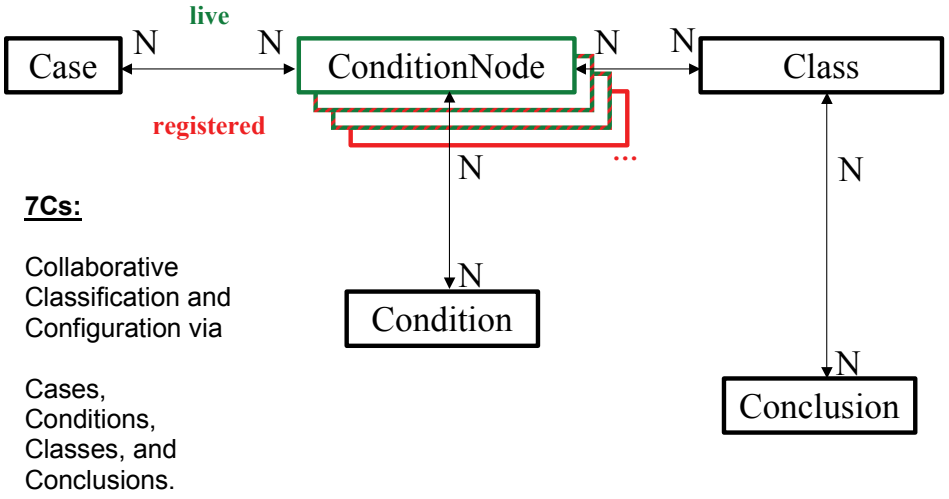
9.2 7Cs Condition Mesh

In the proposed 7Cs structure, each parent ConditionNode (i.e. RuleNode) may have multiple child ConditionNodes, and each child ConditionNode may have multiple parents. A parent node with multiple child nodes supports both a subsumption and a polymorphic exception hierarchy; whereas a child node with multiple parent nodes supports the notion of multiple inheritances, as in Frame-based or Object Oriented class-based domain modelling systems. Kang (1995, p62) has previously highlighted the importance of allowing classifications to be

re-used so that the system can identify when two or more identical classifications are produced, and report on just one of them. In the 7Cs model, identical classifications being served up in different parts of the same condition (i.e. rule) mesh would warrant the use of a shared child ConditionNode. This feature is described in section 13.4 on page 245.

In the 7Cs system there is an N-to-N[165] relationship between Cases and their live and/or registered ConditionNodes; an N-to-N relationship between the ConditionNodes and the Classifications that they represent[166]; and an N-to-N relationship between the resultant Classifications and their Conclusions. Both the *live* public view, and the *registered* individual (and possibly private) views of the current case-RuleNode associations are maintained by the 7Cs system. The case-RuleNode associations can therefore be in several different states for one or more users: just *live*, both *live* and *registered*, or just *registered*. Figure 19 shows an entity relationship diagram for the system. These concepts are explained in more detail later on (sections 11.3.1 and 11.3.2 commencing on page 184).

Figure 19: 7Cs Entity Relationship Diagram

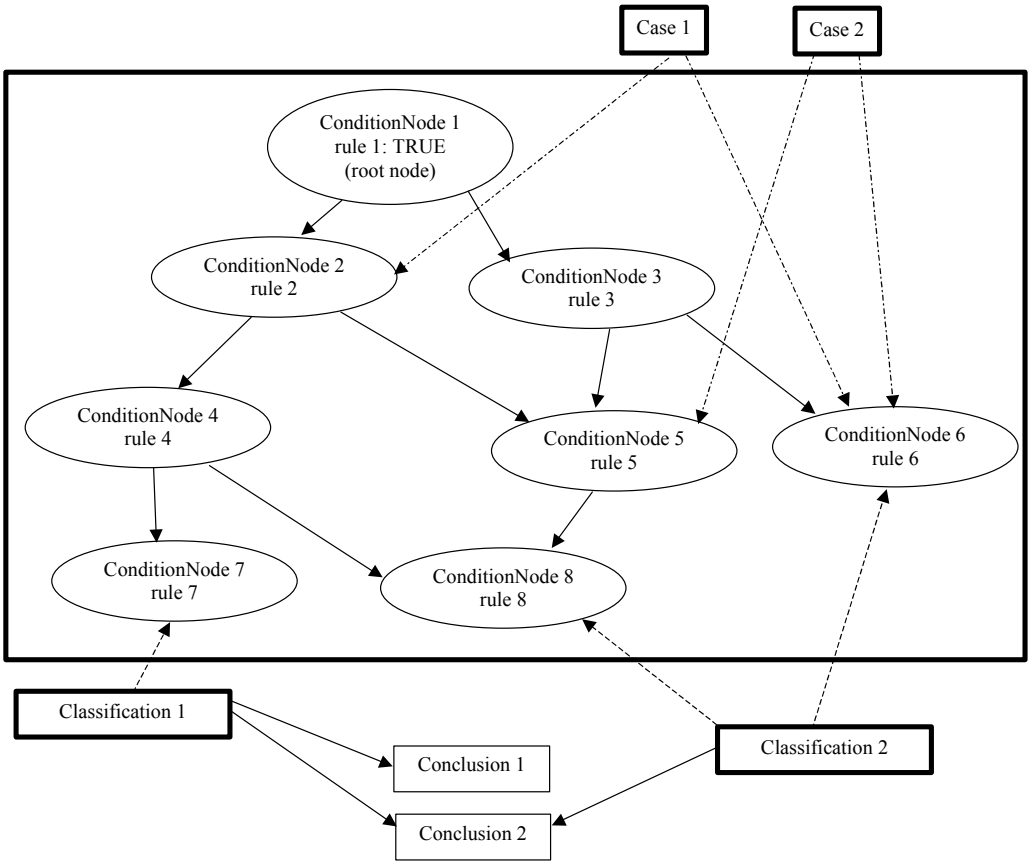


[165] This is a standard nomenclature in relational database design. For example, the N-to-N relationship between Cases and their ConditionNodes means that for every case, there can be multiple ConditionNodes, and for every ConditionNode, there can be multiple cases.

[166] As noted previously (section 7.2.3 on page 97), notionally the mapping between a ConditionNode and the Class that it represents is 1 to 1. In practice it may also be worthwhile to refer to the relevant superclasses at the ConditionNode. This is discussed further in section 11.3.6 (page 193).

An example 7Cs condition mesh is displayed in Figure 20 (the example applies equally well to a live or a registered view of case-RuleNode associations). As shown in Figure 20, a Case can evaluate to `TRUE` for multiple Condition paths in the condition mesh, hence a Case can fetch multiple Classifications, where each Classification may be linked to multiple Conclusions. As well, Conclusions can be reused across multiple Classifications, and those Classifications may be reused across Condition paths, and across multiple Cases. Note that in Figure 20 only a subset of classifications and conclusions are shown for ConditionNodes 6, 7, and 8 and for simplicity links to the classifications and conclusions at other ConditionNodes have not been included in the figure. Further examples of how this structure is useful will become apparent through the remainder of this thesis. Examples include Figure 53 on page 247 and Figure 54 on page 253.

Figure 20: 7Cs Condition Mesh



The 7Cs system allows knowledge workers in any domain to collaboratively refine and expand a topic using an expert systems approach by consistently asking users to confirm, add to, or refine the knowledge presented, typically within the context of a current case. In the system, Attributes, Cases, Conditions, Classifications, and Conclusions can each be the subject of Collaborative editing.

The system goes beyond the collaborative capture of solution knowledge, to promote the collaborative capture and re-use of tacit problem solving knowledge, including classification and problem determination knowledge. The approach adopted does this by capturing and sharing the questions that experts ask themselves when classifying incoming cases.

The system significantly extends the Multiple Classification Ripple Down Rules (MCRDR) algorithm. For example, new data structures and algorithms are presented to facilitate conflict resolution between multiple sources of expertise and to allow experts to more effectively collaborate in building up a knowledge base and condition mesh. The detailed design of the core system is described in Chapter 11 (commencing on page 176). Enhancements to this core design are described in Chapter 13 (commencing on page 242).

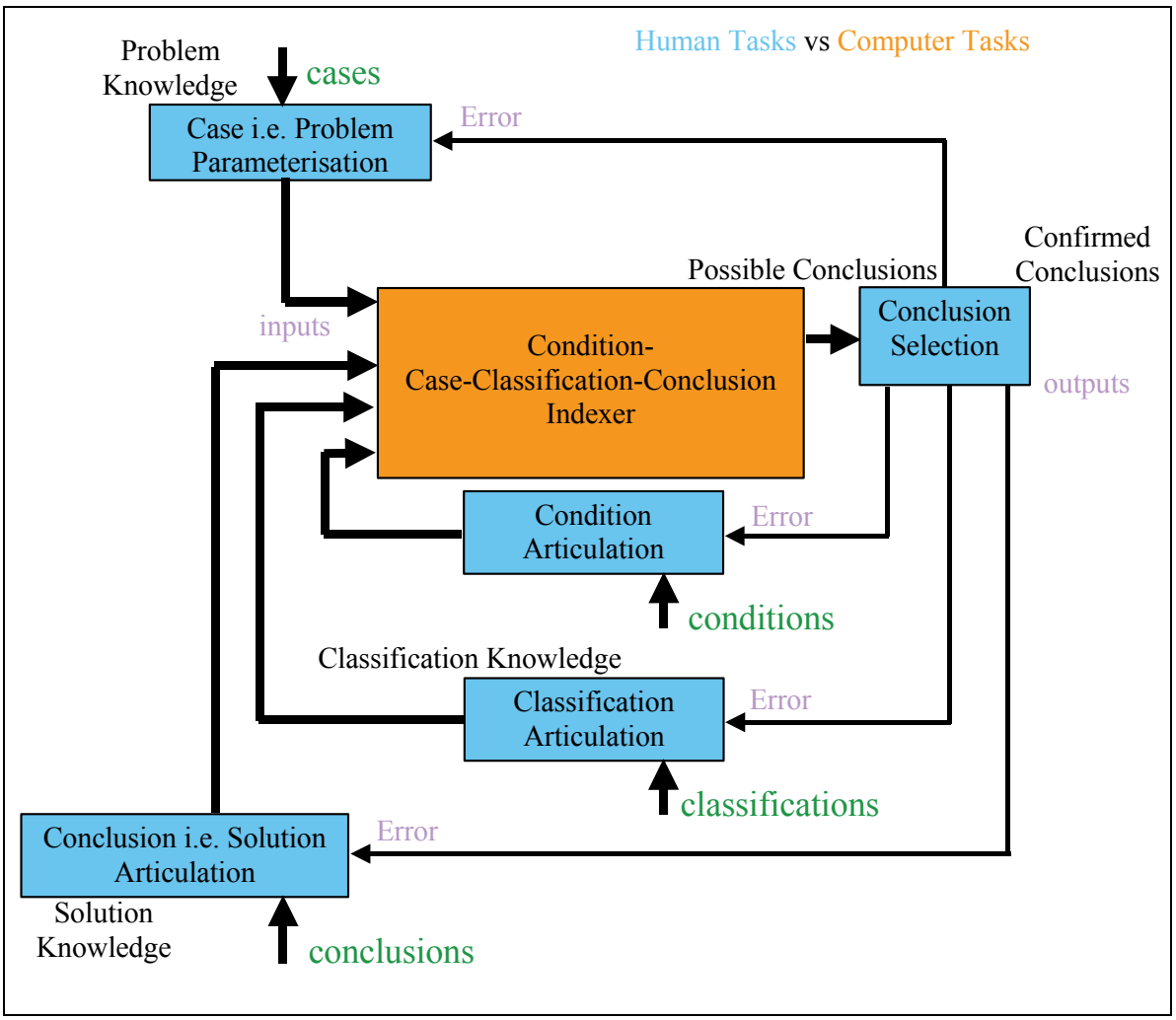
9.3 7Cs Data Flow Diagram

The top-level Data Flow Diagram (DFD) of the 7Cs software architecture is shown in Figure 21. Input arrives in the form of Cases. The cases are parameterised via attribute-value pairs that identify the case. The user is guided to interact with the case recursively to expand its attribute list and thereby configure it to make it explicit enough for the system to identify useful classifications for it. This is similar to the process of working up a case in diagnostic professions such as medicine. As the right questions are asked and more information comes to light the case develops to an extent that a decision regarding the best course of action can be made.

The system identifies classifications for a case by evaluating its attributes against multiple paths of sequential rule nodes in the condition mesh as exemplified by Figure 20. The case will be found by the system to fit with one or more classifications according to the current condition mesh. The system will offer the user one or more conclusions for each classification that the case complies with.

The user will be asked to confirm or deny the presented classifications and their conclusions. Where a given classification is rejected, the user is asked to modify the set of attributes for the case so that it fetches a different classification; edit the classification or its conclusions, or the conditions under which that classification is invoked; or stop an existing ConditionNode and / or add a new ConditionNode in the condition mesh in order to reflect the new classification and its conclusions.

Figure 21: A 7Cs Top-level DFD



In Figure 21 the human's tasks include: problem parameterisation, condition articulation, classification articulation, conclusion articulation and selection of the presented classifications and conclusions; and the computer's tasks include the indexation of conditions, cases, classifications, and conclusions.

Obviously it is vital to the success of the tool that it actually gets used – the more it is used, the more useful it will prove to be. In this vein, multiple users can dynamically update the knowledge.

9.4 Legacy Problem Ticketing System and Knowledge Base

Support centres have made and continue to make enormous investments in evaluating and purchasing workflow software. Quite separate from the financial investment in software is the investment in organizational learning - "*the way we do things around here*". This extends beyond the training of front-line personnel to an investment in custom reports and metrics to assist in performance management of the support centre.

Through the use of hyperlinks (i.e. web-based URIs), the 7Cs model can be used to index cases or solutions in any Intranet or Internet addressable legacy case tracking, solution tracking or other workflow system that the support centre may have already invested in.

9.5 Applicability – Solution Space

In addition to the ICT support centre domain, the 7Cs design lends itself more generally to knowledge domains where users rely on heuristics to form classifications, and where users apply some subset of a reusable set of conclusions depending on the given classification.

In order to achieve a fit with the 7Cs solution, the following pre-condition must be satisfied:

1. The problem must be worth solving i.e. the cost of solving the problem must be less than the cost of replacing the broken unit. For example, suitable problem domains could have:
    - heavy / installed / expensive plant or equipment
    - expensive and / or critical infrastructure such as that provided by the utilities, for example: electricity, gas, water, phone, Internet, or traffic control.

As well, the solution will potentially be of benefit in any of the following situations:

2. Wherever complex problems need repetitively to be solved. For example:
    - support centres, call centres, help desks
    - computer / software / hardware – support / upgrade / compatibility
    - motor / plant / heavy vehicle – mechanics / diagnostic repairers

- health / medical / pathology – diagnosis and prescription
- law, tax, lending, fraud

3. Wherever trouble-shooters find it difficult or require expert knowledge to identify the type of problem on hand. For example:
   - trouble-shooting computing problems in an ICT support centre
   - trouble-shooting problems in an aeroplane
   - trouble-shooting problems in a mining drag-line / loader or other vehicle
4. Wherever trouble-shooters find it difficult to know where to search for information. For example:
   - because there are multiple and sometimes conflicting sources of information
   - because the appropriate place to search for the information depends on the search context
5. Wherever data exists in the form of cases, and custom searches on the individual fields of those cases may be effective in reducing the number of false positives returned by a standard search engine. For example:
   - sifting through hex dumps of error codes produced by faulty software
   - numerical data e.g. blood sugar levels in pathology, needing logical tests such as <, >, !=, <=, >= etc
   - time series or sequential data e.g. animal or plant or organism growth rates
   - data that requires custom pattern matching, for example supporting wildcards in special places.
6. Wherever there are multiple experts with conflicting opinions about how best to identify and solve a problem.
   - Most experts conflict with each other! This system provides a mechanism for highlighting the conflicting cases, and it allows experts to pick out features of a case that differentiate it from past cases and determine that it needs a different solution.

7. Whenever there is a group of users in disparate locations that can benefit from sharing their problem determination, taxonomy, ontology, or classification know-how. Example domains include: botany, zoology, biology, chemistry, pathology, geology, technical analysis in financial markets, and of course trouble-shooting client problems or resolving customer enquiries for example in a support or call centre.

9.6 Chapter Summary

A key feature provided by the 7Cs model in support of collaboration is the separation between the *live* case-RuleNode associations that represent the current global truth calculated and recorded by the 7Cs expert system (i.e. the knowledge base), and the *registered* case-RuleNode associations that represent the current truths of individual contributors to the expert system. In Chapters 11 and 12 we shall see how tracking the difference between live and registered case-RuleNode associations provides for conflict resolution and hence additional knowledge acquisition opportunities, since users can be spontaneously notified of changes to the knowledge base that will affect their past and future decisions.

As well, the 7Cs model lets users collaboratively share the questions they ask themselves when working-up their cases. Contributors are collectively guided by their peers in the problem determination and hence the case configuration process.

In summary, the 7Cs system lets:

- humans do what humans do best: comparative analysis, problem classification, decision making, and solution generalisation; and
- computers do what computers do best: massive and light-speed indexing, repetitive question-answering, significant number crunching to infer solutions, data manipulations to reveal knowledge gaps, and presentation of knowledge structures with multiple views in multiple different and useful ways.

This chapter has presented some top-level concepts for the proposed 7Cs system.

The next chapter describes the specific problem domain explored in the FastFIX software trial.

CHAPTER 10: THE DIAL-HOME PROBLEM CONTEXT

10.1 Chapter Outline

As described previously (Chapter 2), the HTG support centre provides a case study that offers insights to the problem solving process in general. For example, the observations, interview and survey at the HTG support centre confirmed that troubleshooting involves a process of classification and configuration (section 2.5.3 on page 26).

As implied by section D.1 on page 398, information retrieval techniques require that similarity and dissimilarity measures be calculable between items in the search space in order to optimise both the precision and recall of the information retrieved. Similarly, as described in section 3.6 on page 38, Data Mining (DM), Case Based Reasoning (CBR), and Ripple Down Rules (RDR) techniques each require that the user create a model of the domain, comprised of a vector of attributes that can be used to identify each case in the system, and a set of functions or rules that can be used to determine the similarity and dissimilarity between cases in the search space.

This chapter describes the nature of the problem cases being experienced by the HTG support centre in their Hardware Support Lab (HSL). The study of this specific problem context has provided the motivation for the more general 7Cs design and FastFIX implementation as described in Chapters 11 and 13. What we will observe in this problem space is that the similarity and dissimilarity measures between problem cases cannot be discovered by observing or mining the available data alone. Rather, this case-based classification knowledge needs to be acquired from human experts.

10.2 Problem Scope

As mentioned previously, at the time of this research the HTG support organisation was experiencing upward of 5,000 problem cases per day globally. 20% of these cases were arriving in the form of cryptic error codes in hexadecimal format automatically emailed from errant equipment to the support centre's case tracking software in the form of "dial-homes". The remaining 80% of cases were being raised by customers via a support site on the Internet, or by phoning up HTG's customer service desk.

In the last few months of the project, the Sydney-based support group decided to focus the attentions of this research on its dial-home problems[167]. The dial-home cases were handled by a part of the organisation known as the Hardware Support Lab (HSL). When the equipment experienced errors, depending on the seriousness and frequency of those errors an error filtering system inside the equipment would determine whether or not to dial-home and report the errors. Incoming dial-home cases were stored in the CaseDB case tracking system and automatically assigned to a set of incoming case queues to be handled by personnel in the HSL. If additional dial-homes subsequently came in from the same unit of equipment they were added to the (possibly open) unresolved CaseDB case.

Dial-home cases were made up of up to 10,000 different hexadecimal error codes. Much rote-learnt and experience-based memory work was required on the part of experts to interpret them. It was felt by several trouble-shooters in the HTG support centre that an expert systems approach could offer a significant improvement to the process of interpreting and acting on the dial-home cases.

An example case is shown in Appendix B (commencing on p 387). This case is comprised of two separate dial-home events labelled as dh1 and dh2. As can be seen in the last few paragraphs of the case, the case was handled by 11 different support centre personnel and took almost 10 days to be resolved (I have obfuscated the user names to avoid identification of the individuals). Much of the problem solving activity occurred within the first 2-3 days. Hardware Support engineers reported that dial-home cases take on average approximately 15 minutes to solve.

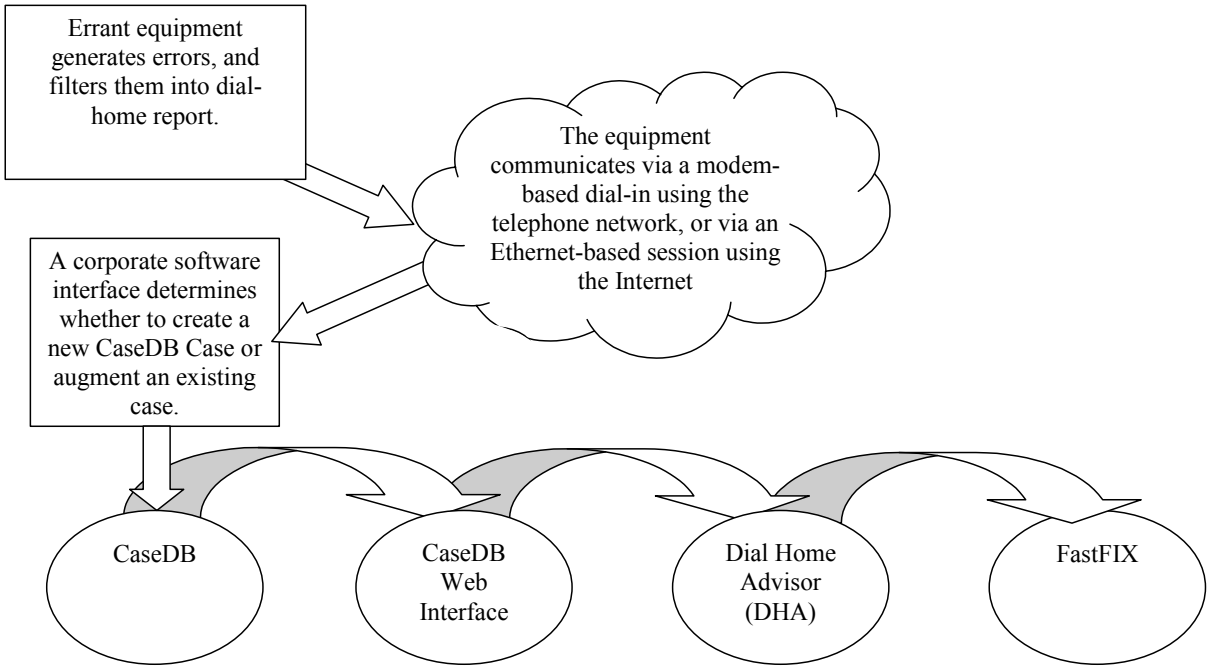
10.3 Dial-home System Overview

The first stage in the system design for the FastFIX dial-home project was to parse the incoming cases and capture important parameters that could be used to determine the class of problem on hand. Given the critical importance of the CaseDB database to HTG globally, I was unable to get data-level read or write access to it. Unfortunately I was also unable to get read or write access to a replica database. However a web-based interface to CaseDB had been implemented that trouble-shooters across the company were able to access.

[167] Many thanks to Stephen Wright, Kieran McGee, Chris Bunting, and Colin Berrell for focussing the project in the area of dial-home cases, and for providing resources for the related FastFIX software trial.

In response, an HTG colleague[168] developed some prototype software that he named the *Dial-Home Advisor* (DHA). The DHA scraped HTML code from the web interface to CaseDB, summarised the important details of any given CaseDB case and passed them to FastFIX for each dial-home in that CaseDB Case. It also gave an initial human readable interpretation of the hexadecimal error codes in the dial-home. The following figure shows the resulting chain of software used to gather, manipulate and communicate the dial-home case-data.

Figure 22: Flow of Case information from the equipment to FastFIX

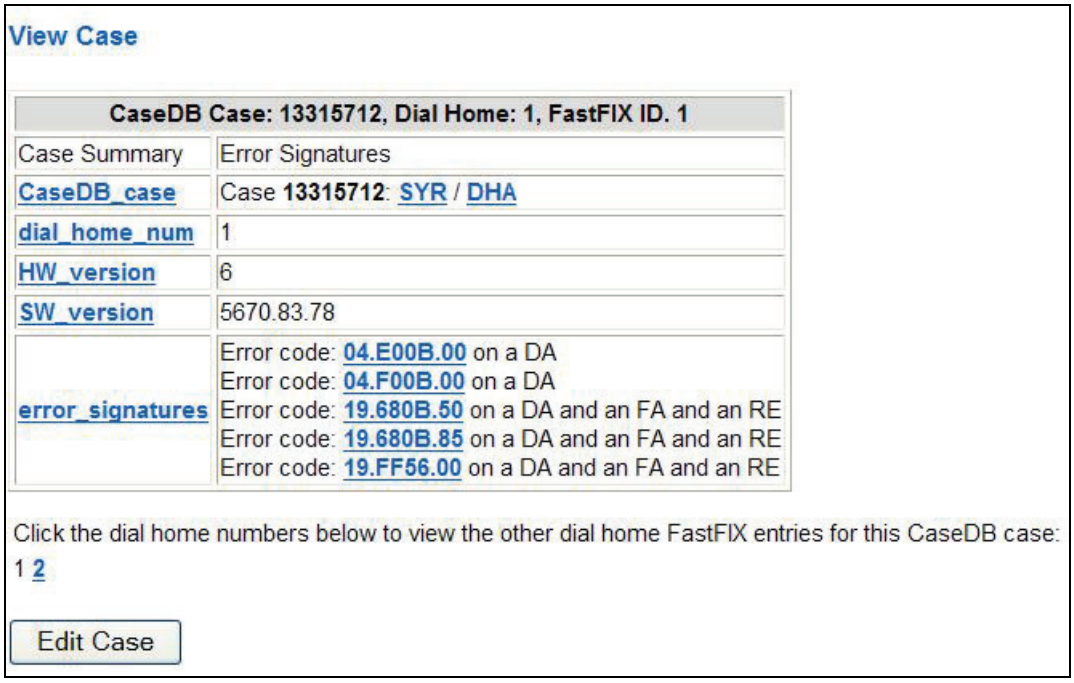


The DHA was implemented in Perl / CGI and could be accessed via a website by trouble-shooters in the HSL. Trouble-shooters simply entered the identity number of the CaseDB Case to the DHA and then for each dial-home included in the case, they were directed to the FastFIX system and its conclusions.

The following figure shows a screenshot from the FastFIX system of the first dial-home for CaseDB Case ID 13315712 as shown in Appendix B (commencing on p 387).

[168] Many thanks to Chris Bunting at HTG for generous assistance with understanding the nature of the HTG dial-home problems, and for conceiving of and developing the necessary DHA interface software.

Figure 23: Case Data for FastFIX ID 1: CaseDB ID 13315712 – Dial Home 1



View Case

CaseDB Case: 13315712, Dial Home: 1, FastFIX ID. 1	
Case Summary	Error Signatures
CaseDB_case	Case **13315712**: SYR / DHA
dial_home_num	1
HW_version	6
SW_version	5670.83.78
error_signatures	Error code: 04.E00B.00 on a DA Error code: 04.F00B.00 on a DA Error code: 19.680B.50 on a DA and an FA and an RE Error code: 19.680B.85 on a DA and an FA and an RE Error code: 19.FF56.00 on a DA and an FA and an RE

Click the dial home numbers below to view the other dial home FastFIX entries for this CaseDB case:
1 2

Edit Case

For the CaseDB Case and dial-home shown in shown in Figure 23, the parameters passed to FastFIX by the DHA included the identity of the CaseDB Case, the identity of the dial-home, and a summary of the error signatures for the dial-home.

The ability to have more than one dial-home linked by the same CaseDB ID is similar to the ability in PEIRS to have more than one specimen or test result linked to the same patient ID. In PEIRS, interpretations could be made both for the individual test results, or for the combined suite of test results for a given patient (Edwards, 1996, pp 81, 100).

As mentioned earlier, there were up to 10,000 different valid error codes. The meaning of the error code format for HTG dial-home cases is as shown in Table 11.

Table 11: Meaning of the LL.CCSS.mm error code format for dial-home cases

'LL'	The general error code category.
'CC'	The 1-byte context code which could be the identity of the channel/scsi command or the utility/test that was running when the error occurred, or something else depending on the nature of the error.
'SS'	The 1-byte symptom code, ie the main classification of the error.
'mm'	The modifier which gives a more specific definition of the error and is primarily for engineering use.

In addition, each error code could be seen on one or more of 10 different types of hardware "directors". The following table shows the different types of hardware directors:

Table 12: Available director types

'RA'	Escon RDF Director
'FA'	Fibre Host Adaptor
'EF'	Ficon Host Adaptor
'EA'	Escon Adaptor
'RF'	Fibre RDF Director
'EFLINK'	EF Link Processor
'RE'	GIGE RDF Director.
'SE'	iSCSI GIGE Host Adaptor
'DA'	Disk Adaptor
'SW'	MSWindowsSimulator

As well, the error codes could have one or more qualifying *sense bytes* attached to them, each with their own unique values.

For example in the FastFIX rule syntax, XX.FF56.XX:S12=01&S13=02:DA refers to an error code that matches XX.FF56.XX where X means 'don't care' and can be any hexadecimal character (0-9 or A-F); S12=01 means that sense byte 12 has a value of 0x01; S13=02 means that sense byte 13 has a value of 0x02, and DA means that the error occurred on a DA type hardware director.

In contrast to the parameters passed from the DHA to FastFIX, the hardware version and the software version shown in Figure 23 on page 172 had to be manually supplied by the trouble-

shooter by editing the case. At the time of deployment there were 7 different versions of hardware in the field, and many more different software versions.

From the above description, the following characteristics of the dial-home problem domain become clear:

- there was a huge number of possible permutations and combinations of error;
- there was a high reliance on memory and knowledge from a wide variety of experts and sources;
- a good deal of parameterised case data was already available for incoming cases;
- much of the data was numeric, and the data lent itself to complex numeric and string-based evaluations;
- the data did not lend itself to simple keyword searches; and
- several of the parameters pertinent to solving the case needed to be manually supplied.

As well as this and perhaps most significantly:

1. there was a high repetition rate for certain classes of problem which meant that automating the trouble-shooting process for those problem classes would be very worthwhile for the HSL and for HTG more generally.

The numeric and non-searchable properties of the dial-home data are similar to the properties of the pathology data reported by Edwards in PEIRS (1996, p69).

10.4 Chapter Summary

Information retrieval (IR) and Artificial Intelligence (AI) techniques require that similarity and dissimilarity measures be calculable between cases in the search space. These techniques require a model of the domain, comprised of a vector of attributes that can be used to identify the cases, and a set of functions or rules that can be used to determine the similarity and dissimilarity between the cases. What we find in the HTG HSL case study is that the similarity and dissimilarity of problem cases cannot be determined by observing the available data alone. Rather, this case-based classification and configuration knowledge needs to be acquired from human experts.

Although the HTG HSL case study has motivated the design of the 7Cs solution proposed in this thesis, 7Cs has more general applicability. 7Cs was designed to address information retrieval problems in any case-based problem domain where classification and configuration knowledge needs to be acquired from one or more human experts, or users. The acquired configuration knowledge is used to work-up the problem cases, and the acquired classification knowledge is used to calculate the similarity and dissimilarity between them. This guides users in their retrieval of information in that problem domain, automates the information retrieval process, and improves the precision and recall of the information retrieved.

The next chapter presents the detailed design of the 7Cs and FastFIX system.