TROUBLESHOOTING AT THE CALL CENTRE: A KNOWLEDGE-BASED APPROACH

Megan Vazey and Debbie Richards
Computing Department
Macquarie University
megan@excelan.com.au, richards@ics.mq.edu.au

ABSTRACT
The key focus of the customer call-centre is effective and efficient resolution of customer problems. Keeping staff and clients happy by streamlining call-centre workflow is integral in achieving this end. We propose to extend a knowledge representation and acquisition technique, known as multiple classification ripple down rules (MCRDR), that will support management of troubleshooting knowledge from multiple sources, including in-house databases of past and current cases and relevant internet-based material. We present a prototype system, and a database design that seeks to align the goals of the call-centre, such as maintainability and workflow compatibility, and which will inter-operate with the call-centre's existing problem ticketing and knowledge management systems.

KEY WORDS
Knowledge acquisition, knowledge management

1. Introduction

Much has changed in the last 15 years in regard to information systems. To begin with, incredibly rich and globally accessible internet content has shifted our focus from bookshelves and libraries, to search engines and hyperlinks.

Despite the information revolution heralded by client-server internet technology, the problem for the call-centre help-desk / service-desk remains the same: how can we achieve rapid access to the minimum set of knowledge that will help us solve the problem on hand?

Vendors and technologists alike have grappled with this problem. Techniques from the field of artificial intelligence and data mining such as case-based-reasoning, genetic algorithms, neural networks, clustering, and nearest neighbour algorithms have been applied. While these techniques may work well in a stand-alone and static knowledge environment, we argue that in the often incomplete and dynamic knowledge environment presented by the call-centre, integration with the way people work, their workflow, and the natural incentives that motivate people to use a system, is necessary for system success.

We propose that the Multiple Classification Ripple Down Rules (MCRDR) technology initially introduced by Kang, Compton and Preston [1] to the pathology domain will fit well in this domain and will offer an effective workflow solution. A key differentiating feature of this technology is that it offers a closed-loop feedback system where-by users continually update and refine system search results, and hence the system knowledge as part of their daily work effort. A self-maintaining expert system is therefore presented.

This paper has three main sections. Firstly, we provide a brief review of existing vendor and case-based reasoning solutions for knowledge acquisition and reuse at the call-centre and help-desk. We then provide an introduction to MCRDR technology, and we present previous applications of the technology to the help-desk domain. Next, we present our system architecture and the extensions we have made to MCRDR. Finally, we offer our conclusions and next steps.

2. In Search of Solutions

In this section we take a brief look at existing vendor solutions, and we review the manner in which case-based reasoning technology, of which MCRDR can be regarded a subset, has been applied to the domain.

2.1 Vendor Solutions

We are not the first to attempt to help the help-desk. Countless software vendors promote a wide variety of knowledge management solutions. For instance, http://www.helpdesk.com (2004) lists: 314 vendors of "Help Desk" software aimed at automating the help / service desk; 26 vendors of "Knowledge Management" software including document management, collaboration and knowledge sharing, and search and categorization tools; 133 vendors of "CRM and Call Centre" software that helps automate the call centre and customer management process and 7 vendors of "Defect Tracking" software.

Vendor solutions to knowledge management at the help-desk include case-based and rule-based reasoning systems, collaborative forum software, and knowledge structuring tools such as FAQ builders. Established and emergent technologies include clustering algorithms, neural networks, and genetic algorithms. A significant number of these vendors use expert system technology to assist with knowledge management.

As Call-centre and help / service desk software is squarely aimed at building the corporate-client

relationship through enhanced customer-centric communications we have found that much of what is offered as call-centre / troubleshooting / case tracking systems are sophisticated databases that are able to keep the current status of the case up-to-date and the customer informed. In most cases the systems do not assist with decision-making. Instead the focus is on decision tracking and reporting.

Some approaches incorporate techniques from artificial intelligence such as: classification and decision trees, fuzzy logic, artificial neural networks and genetic algorithms. Many of these techniques are used for data mining purposes to develop association or classification rules. While we may incorporate such techniques further down the track if deemed appropriate, data mining is not our key interest. Our key motivation for seeking an alternative is that in the call centre environment the knowledge, and the cases which provide the context for the knowledge, are changing. Additionally, we want to make extensive use of external sources of knowledge, and knowledge locked up in existing corporate databases, to assist in the problem solving process. Our interest in maintaining a case-base in addition to a knowledge-base led us to review research in the field of case-based reasoning.

2.2 Case-Based Reasoning (CBR) approaches

The potential value of CBR for help desk applications has been recognized by numerous researchers. Following on from the initial work of Roger Schank in the early 80s, numerous CBR systems have been developed for the Help-desk (e.g. SMART [2], CASCADE [3] and CARET [4]). Kim et al. [5] note that the methods that CBR systems use to index, compare and modify cases, necessitate a degree of knowledge engineering and that Help Desk Systems (HDS) exist in dynamic environments which become susceptible to maintenance problems.

An example of a typical CBR approach, which was also in the Help Desk domain is Kriegsman and Barletta [6] who used a symbol hierarchy to assist in the organization and retrieval of cases and their important features. A nearest neighbour algorithm (NNA) was applied to rank the similarity of cases within a class and a symbol hierarchy was needed to determine similarities between classes.

The PROTOS system [7] is a well-known CBR method that shares a number of characteristics in common with RDR. PROTOS is a failure-driven approach, which uses surface features of a new case to identify categories that it may belong to. A similar case, called an exemplar, from the category is selected. The exemplar is compared to the new case. If the exemplar is not similar enough, difference links between cases are identified to assist choosing a new exemplar. The use of cornerstone cases and difference lists in RDR can be compared to the exemplar and difference links in PROTOS. However in RDR, cornerstone cases aim to confirm or modify the classification given and to use the differences between the cornerstone/s and current case to create an index to the current case for future inferencing.

In both PROTOS and RDR, learning involves feature bias and both allow cases to be augmented with new features. Just as we have found in mainstream ontology and KA research, PROTOS relies on development of a "good" model in order for its matching process to be successful.

When a domain theory is not available to use in matching, if the terms in the model do not have an agreed upon definition or when there are competing models, suitable matches will not be found [8]. On the other hand RDR is a simple technique that requires minimal *a priori* modeling, but which classifies and indexes cases while experts exercise their expertise.

3. Introducing MCRDR

In this section we introduce the MCRDR Decision Tree, we review the manner in which MCRDR has been applied to the Help Desk domain, we summarise the key characteristics of MCRDR systems, and we discuss the challenges that this domain presents to previous implementations of MCRDR.

3.1 The MCRDR Decision Tree

The MCRDR algorithm [1] is now a decade old and in that time it has seen numerous implementations. In a programmatic sense, the algorithm is clearly and concisely explained in its founding paper, however, for readability we include our own brief explanation of the MCRDR decision tree.

As shown in Figure 1, an MCRDR decision tree is:

(i) An N-ary Tree of RuleNodes.
(ii) Each RuleNode has a *rule* and a *conclusion*.
(iii) The topmost RuleNode in the tree evaluates to TRUE for every case in the system.
(iv) Cases comprise of *attribute-value* pairs for example, a customer's software problem could be described using the following attribute-value (A-V) pairs, *'software version' == '5.1'*, *'operating system' == 'winXP'*.
(v) The *rule* at each RuleNode tests one or more feature(s) of the case's attribute-value pairs for example, that *'operating system' == 'winXP'*. Each case is evaluated against the topmost parent RuleNode and then successively down the tree for each child RuleNode.
(vii) If the result is TRUE for a parent Rulenode, the case is recursively evaluated against all of its child RuleNodes.
(viii) The live conclusion list for a Case includes the conclusions from the last TRUE RuleNode in every path down the RuleNode Tree.

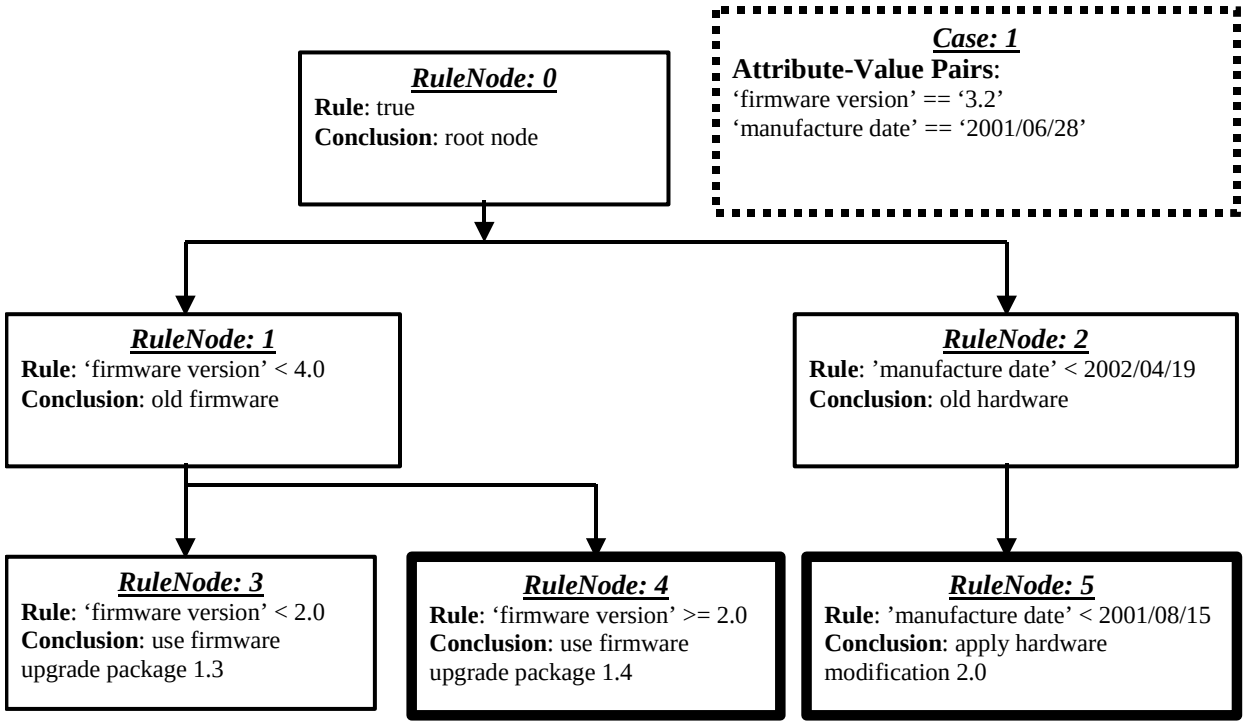

Figure 1: MCRDR Decision Tree (Case == Hardware Fault)

As an example, Case 1 in Figure 1 describes the case of a Hardware Fault where ('firmware version' == '3.2') and ('manufacture date' == '2001/06/28'). The last TRUE RuleNode in every path down the RuleNode Tree gives the conclusions from RuleNodes 4 and 5: 'use firmware upgrade package 1.4', and 'apply hardware modification 2.0'. When a new case is added to the system, the user can choose to accept a given conclusion or alternatively reject it by creating a differentiating rule with an alternate conclusion. In that case:

(i) The new rule must be a valid boolean expression which is able to be evaluated by the MCRDR engine. The rule for the new RuleNode should be different from the rules of its ancestor RuleNodes.

(iii) The rule for the new RuleNode may optionally be restricted to a single test, for example, that ('firmware version' >= 2.0), rather than a conjunction of tests[1].

(iv) The new RuleNode must have either a different conclusion, or a different rule compared to its sibling RuleNodes.

(v) The new RuleNode must test for some feature of the Review Case and must evaluate to TRUE for the Review Case.

(vi) The new RuleNode must distinguish between the Review Case and all of the Cornerstone Cases for the parent RuleNode.

New RuleNodes can generally[2] be placed at one of two places in the tree [1]:

(i) At the top of the RuleTree to provide a new independent conclusion.

(ii) Beneath the current RuleNode as a replacement conclusion or as a stopping conclusion.

3.2 MCRDR and the Help Desk Domain

MCRDR has been previously explored in the help desk environment [9, 5, 10].

The prototype described in [9] 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. Their system considered updates by a single expert only. As noted in [9] 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 in [9] 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 cases and update the knowledge accordingly. The idea is interesting and may prove to be a useful adjunct for browsing the knowledge in our system.

[1] Where more complicated conjunctions of tests are allowed, the new RuleNode is more likely to be added to the top of the tree and a stopping rule used at the end of the path – the overall result is a flatter rule-tree structure (Kim, 2003).

[2] Actually, there is a possibility that new RuleNodes could be placed in the path between the topmost RuleNode and the current RuleNode by asking the user to identify the minimum set of rules in the current path that the case must satisfy for the new RuleNode.

The prototype in [5] extended the prototype in [9] by allowing an expert user to also build and maintain the help desk document knowledge base by applying keywords to help documents.

In her PhD thesis, Kim [10] applied the concept lattice from Formal Concept Analysis (FCA) [11] to generate a browsing structure to assist users in navigating the knowledge base.

3.3 Key Characteristics of MCRDR systems

Various implementations of the MCRDR and single classification RDR algorithm have been developed over the past fifteen years which indicate the versatility of the RDR structure. Generally, each variation used:

- some form of pre-processing of the raw data;
- cases to provide context, to assist in forming rules, and for validation;
- a simple model comprised of A-V pairs and conclusions;
- incremental KA and maintenance; and
- the exception structure.

The differences between the implementations tended to concern how the knowledge was being presented and manipulated and the inferencing strategy. The key benefits provided by the RDR approach, resulting in its commercial success in the pathology domain [12, 13] lies in it being a technique that can be easily adapted to fit with current practices in an organisation and that the knowledge can be maintained by domain experts without the mediation of a knowledge engineer. The 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 provide context and the exception structure to support local patching of the knowledge. It is these features that make MCRDR an attractive basis for the acquisition and maintenance of Call Centre knowledge.

3.4 Challenges for MCRDR

Despite the attractive features outlined above, the call-centre help-desk context under consideration has a number of properties that present new challenges for the MCRDR algorithm:

- the system must interact with a legacy ticketing system and legacy knowledge base
- the system needs to deal with numerous cases (in the order of 50 per day locally, and 300 per day globally)
- the volume of cases being dealt with means that the workflow must inherently deal daily with system maintenance and knowledge acquisition
- initial problem descriptions are sparse – the case definition matures as the customer service personnel interacts with the customer and *works 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 (up to a day) between when the system receives a problem case, and when a customer service representative can attend to it.
- 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
- multiple users will use and update the system, but a limited subset of privileged users will approve their updates
- 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.
- The range of values possible for those attributes is also very large and the dependencies between these A-V pairs may be very strong. For example, we are dealing with troubleshooting across multiple systems, platforms, vendors, versions, etc.
- We don't have control over the cases, which are stored in the parent company's database.
- The A-V pairs and rules in our system are not 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'.
- Multiple users will describe the cases through an interactive question-answer interface to the system that will assign the relevant A-V pairs to the case.
- Our system will be maintained by multiple users, not just a single user
- Our 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.

4. Our System

In our system, incoming calls are logged in a legacy call / defect tracking database. Basic features of the incoming case are logged such as date, time, client name, and query summary. More specific details may also be included such as the name, model and / or version of any defective product (e.g. hardware or software) together with a query description.

Our system will allow users to record, retrieve, review, refine and rate troubleshooting knowledge in the context of specific problem classes stored as cases in the legacy defect tracking system. We aim to capture the minimum set of knowledge required to solve customer problems.

When a new problem ticket comes in, the customer service personnel will be presented with a set of refinement queries enabling them to more specifically describe the type of problem being observed by the customer. Immediately that the new information is

entered, the history of how *similar problems were solved in the past* will be presented to the user such as which internet links proved useful, and which legacy knowledge-base references helped. The user will then be prompted to refine the system's knowledge in the context of the given problem class.

Figure 2 shows the top-level architecture of our prototype system. We allow multiple intranet clients to access the system via their web browsers. Our PHP server side code serves up HTML and Javascript to run in the client's browsers.

The MCRDR decision tree is stored in a MySQL system and provides http internet hyperlinks to Cases in a legacy defect tracking database, and Solutions in a legacy knowledge base. At this stage our system is a prototype and we are yet to evaluate the usability and performance of our system architecture and design.

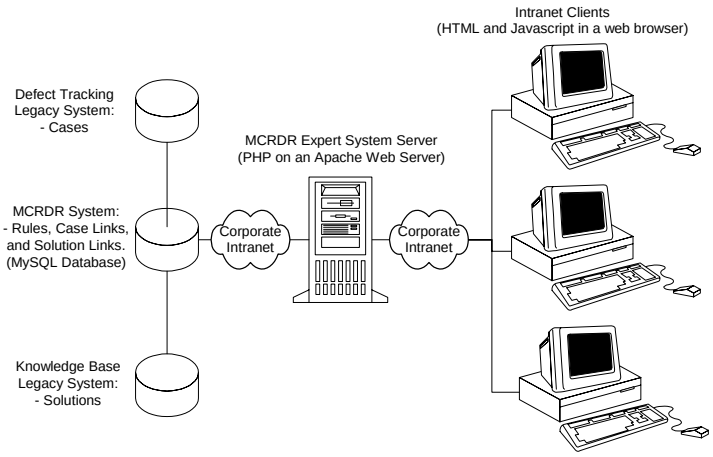

Figure 2: System Overview

Figure 3 represents a high-level entity relationship (ER) diagram that summarises the relationships between the data structures used in the MCRDR database for our system. Referring to Figure 3 Cases contains history of case statements that have been added to the case over time by users. Cases also contain a list of Attribute-Value (A-V) pairs where each attribute has a name; a type (for example: one-of-a-set, some-of-a-set, float-with-range, integer-without-range, or free-text); a set of accepted-values (as required by one-of-a-set, some-of-a-set or ranged attribute types), the attribute display units (for example kilograms, metres); the order in which the attribute should be displayed relative to other attributes; and the System Log or History showing who created the attribute, who modified it, and when these events occurred.

A key aspect of our system is that the structure provides for a many to many relationship between Cases and RuleNodes via the conceptual use of the Registered RuleNode List, Approved RuleNode List, and Live RuleNode List stored with each case; and via the Registered Case List, Approved Case List, Live Case List, and Live Path List stored with each RuleNode. The notions of Registered, Approved and Live cases is novel to our approach and have been developed to address the fact that in the Call Centre domain not only is the knowledge changing but also the cases.

A RuleNode Assoc(iative) List in the Case Table, and a Case Assoc(iative) List in the RuleNode Table, together with the Case-RuleNode Association Table, provide an alternate implementation that is designed to allow for more effective management of the change history of the associations between Cases and RuleNodes.

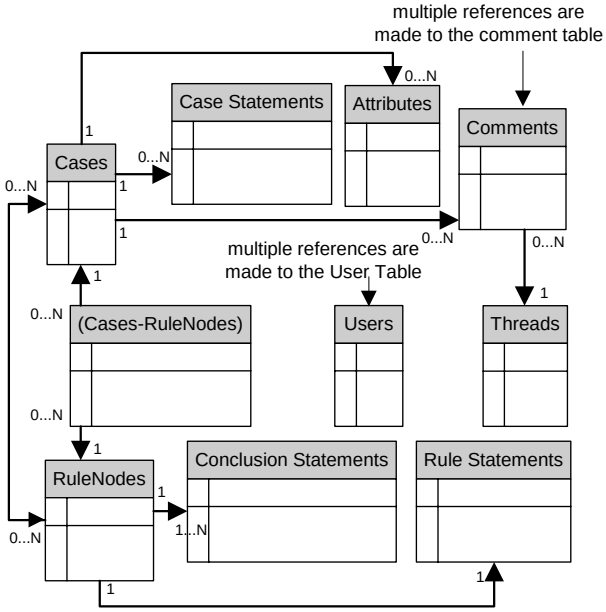

Figure 3: MCRDR Database High Level Entity Relationship Diagram

In keeping with the MCRDR decision tree, each RuleNode contains a reference to its parent RuleNode, its child RuleNode if one exists, and its sibling RuleNode if one exists. RuleNodes also contain a record of their cornerstone cases (which must be both live and registered). RuleNodes contain a Rule Statement, which is a boolean expression that can be evaluated by the MCRDR engine to determine the truth of that rule for a given case. RuleNodes also contain a set of Conclusion Statements where each conclusion statement can be an internet URL, plain text, or an instruction to the MCRDR engine to interactively prompt the user for more A-V details and then recursively re-evaluate the case. Our design provides for multiple RuleNodes to refer to a given Conclusion Statement.

We provide RuleNode and Conclusion confidence scores by analysing the usefulness rating (0 to 5) assigned by users (this mechanism may be augmented to provide a confidence in the context of a given RuleNode).

Cases, RuleNodes, Attributes, and the system itself can be commented on and those comments can be organized into internet forum-style threads.

Users are given a username and password, their job role is identified, and counts are kept of the number of cases that they have augmented, the number of cases they have closed, and the number of RuleNodes that they have created. Together with an indication of how long users have been using the system, these counts are used to assign users with an overall credibility score.

One idea is to link staff incentives to the User Credibility Score, for example by giving out movie tickets for "gold users" – those that provide the most used and highest rating conclusions and / or RuleNodes.

Our design provides for the possibility that Cases, RuleNodes, Rule Statements, Conclusions, Case-RuleNode Associations, and Attributes may be all the subject of asynchronous editing. It remains to be seen whether such an approach will overwhelm users with choice. In that case, we may scale back some of the flexibility to restricted use by the system's Administrators.

5.Conclusion

In this paper we have focused on introducing the prototype system we have developed for the call-centre domain in which we are working. We recognise that success will require a socio-technically balanced solution and have developed strategies to address feedback and collaboration, managing incentives to users to encourage the entry of good conclusions, the handling of conclusion granularity and expiration, accessibility to the system globally and continuously and usability. While we do not have space here to discuss our proposed strategies to these issues, the RDR philosophy is very user-centred, unlike many expert system approaches, and supports well the need to adapt to the organisational culture.

Nevertheless, the call-centre does pose numerous challenges for existing MCRDR implementations. Due to the features of this domain, particularly the evolving nature of the cases themselves, we have suggested a number of modifications to standard MCRDR and have proposed Interactive Recursive MCRDR. These modifications are explored in more detail in [14].

We are currently integrating the prototype with the organisation's databases and problem ticketing systems in order to conduct a number of investigations to determine the robustness of our approach, ranging from evaluation of the algorithm through to system performance and usability, particularly addressing the handling of multiple users updating the system. As well, we intend to explore ways to motivate users to close the loop on system searches and provide the feedback that is essential to MCRDR knowledge refinement.

6. References

1. Kang, B., P. Compton and P. Preston (1995). Multiple Classification Ripple Down Rules: Evaluation and Possibilities. Proc. 9th KAW, Banff, Canada, Uni. of Calgary.
2. Acorn, T. and Walden, S. (1992) SMART: Support Management Automated Reasoning Technology for Compaq Customer Service. In Proc. of the 4th Innovative Application, of Artificial Intelligence Conference. 1992.
3. Simoudis, E. (1992) Using case-based reasoning for customer technical support *IEEE Expert* 7(5): 7-13.
4. Shimazu, H., Shibata, A. and Nihei, K. (1994) Case-Based Retrieval Interface Adapted to Customer-Initiated Dialogues in Help Desk Operations. AAAI'94, pp:513-518.
5. Kim, M., Compton, P., Kang, B. (1999) Incremental development of a web-based help desk system. Proc. of AKAW 99), UNSW, Sydney, 5-6 Dec 1999, 13-29.
6. Kriegsman, M. and Barletta., R. (1993) Building a Case-Based Help Desk Application, IEEE Expert 8(6) Dec. 1993: 18-26.
7. Bareiss E. R. (1989) Exemplar-Based Knowledge Acquisition: A Unified Approach to Concept, Representation, Classification and Learning Academic Press, Boston.
8. Kolodner, Janet (1993) *Case-Based Reasoning* Morgan Kaufman Publishers Inc., San Mateo, CA.
9. Kang, B., Yoshida, K., Motoda, H., Compton, P. and Iwayama, M., (1996) A help desk system with intelligent interface in P. Compton, R Mizoguchi, H Motoda & T Menzies, PKAW'96 Sydney, UNSW, pp 313-332.
10. Kim M. (2003) PhD thesis, Document Management and Retrieval for Specialised Domains: An Evolutionary User-Based Approach. UNSW, 2003.
11. Wille, R. (1992) Concept Lattices and Conceptual Knowledge Systems *Computers Math. Applic. (23)* 6-9: 493-515.
12. Edwards, G. Compton, P., Malor, R., Srinivasan, A. and Lazarus, L. (1993) PEIRS: a Pathologist Maintained Expert System for Interpretation of Chemical Pathology Reports *Pathology* 25:27-34.
13. Lazarus, L. (2000) Clinical Decision Support Systems: Background and Role in Clinical Support, http://www.pks.com.au/CDSS_White_Paper_doc.pdf
14. Vazey, M. and Richards, D. (2004) Achieving Rapid Knowledge Acquisition, *Proc. PKAW'04*, Kang, B.H., Hoffman, A., Yamaguchi, T. and Yeap, W. K. (eds) in conjunction with The 8th Pacific Rim International Conf. on Artificial Intelligence August 9-13, 2004, Auckland, New Zealand, 74-86.