AN EXPERT SYSTEM METHODOLOGY FOR SMEs AND NPOs

Richard Dazeley
University of Ballarat
P.O. Box 663,
Ballarat, Victoria, 3353,
Australia.
Tel: +61 (03) 53 27 9769;
Fax: +61 (03) 53 27 9289;
E-mail: r.dazeley@ballarat.edu.au

Abstract

Traditionally Expert Systems (ES) require a full analysis of the business problem by a Knowledge Engineer (KE) to develop a solution. This inherently makes ES technology very expensive and beyond the affordability of the majority of Small and Medium sized Enterprises (SMEs) and Non-Profit Organisations (NPOs). Therefore, SMEs and NPOs tend to only have access to off-the-shelf solutions to generic problems, which rarely meet the full extent of an organisation's requirements. One existing methodological stream of research, Ripple-Down Rules (RDR) goes some of the way to being suitable to SMEs and NPOs as it removes the need for a knowledge engineer. This group of methodologies provide an environment where a company can develop large knowledge based systems themselves, specifically tailored to the company's individual situation. These methods, however, require constant supervision by the expert during development, which is still a significant burden on the organisation. This paper discusses an extension to an RDR method, known as Rated MCRDR (RM) and a feature called prudence analysis. This enhanced methodology to ES development is particularly well suited to the development of ES in restricted environments such as SMEs and NPOs.

Keywords: Expert Systems, Knowledge Based Systems, Decision Support, Small and Medium Enterprises, Non-Profit Organisations, Prudence Analysis

1. Introduction

Expert Systems (ES) technology has had a resurgence due to the application of business rules within government departments and large organisations. These entities are able to supply the capital required by ES development companies to build customised solutions. Unfortunately, this resurgence has largely overlooked small to medium-sized enterprises (SME) and non-profit organisations (NPO), due to the prohibitive development costs of ES which these organisations cannot meet on their own. This does not mean there are not ES based applications available for SMEs and NPOs, but they tend to be *off-the-shelf* solutions to generic problems.

General ES that solve common business problems, such as scheduling and stock control are obviously required. However, each individual organisation will sometimes also have a number of non-generic problems that are specific to them. Therefore, without being of a size large enough to have custom software built, organisations are left to find non-technological solutions or simplistic methods such as creating a Wiki. This inability of SMEs and NPOs to capture, formalise and record the knowledge of their experts adds significant problems should one of them leave the organisation taking the knowledge with them. This inherently disadvantages the organisation against larger competitors.

The primary problem in ES development is the difficulty in acquiring the knowledge for the system. Traditionally this has required a knowledge engineer to perform extensive modelling through extended expert consultation. This consultation process causes disruptions within an organisation as key personnel are repeatedly taken from their normal practices. In a small organisation the cost of the knowledge acquisition process is two fold: the direct costs of the knowledge engineer and developers' time; and, the indirect cost of losing key personnel for extended periods.

This paper discusses the issues involved in ES development and how they prevent SMEs and NPOs from utilising such systems. Subsequently it will discuss a family of methodologies, based around Ripple-Down Rules (RDR) [1], which bypass these traditional difficulties, plus introduce a new approach, Rated MCRDR (RM) [2] using prudence analysis (PA) to further improve the development process. This final methodology allows SMEs and NPOs an affordable means for developing custom solutions to their own unique business practices. The first section will discuss the problem with current methodologies in ES development for SMEs and NPOs. The subsequent section will provide a basic description of RDR, Multiple Classification RDR (MCRDR) and Rated MCRDR (RM). Lastly this paper discusses the advantage of using prudence analysis, for the development of systems without a knowledge engineer.

2. Problems with Traditional Expert System Methodologies

The traditional expert systems' view of knowledge was founded on the *physical symbol hypothesis* [3], which takes the view that knowledge is comprised of symbols, and the connections between those symbols, representing pieces of reality. Furthermore, that intelligence comes from the appropriate manipulation of these symbols and relationships. The result of the physical symbol hypothesis was that Expert System researchers assumed that such symbols and relationships should be extractible and usable without any further need of the expert. This is the foundational argument behind Expert System techniques.

However, the process of extracting the knowledge from an expert's mind is extremely difficult and time consuming. Numerous methodologies have been developed over the years to aid the knowledge acquisition process. The most well known of these is Knowledge Acquisition and Design Structuring (KADS) [4], which is part of a group of techniques referred to as *task oriented* methodologies. KADS uses an array of modelling techniques, where the expert and knowledge

engineer (KE) work together to build up a set of tasks so that knowledge acquisition can be approached in a systematic way [5].

These task oriented methodologies have been very effective in eliciting the required knowledge in a number of problem domains. However, the process is extremely complex and the KE is a crucial element. They are required to work closely with the expert to first build the tasks and subsequently acquire the knowledge. The expert will also need to dedicate massive amounts of time to the development of the final system. This, however, is a significant problem because the nature of an expert is that they hold vital knowledge that others do not, which makes them difficult to replace while they work with the KE. While this is difficult for large companies they generally have other people with sufficient knowledge to fill in for the expert. However, for SMEs and NPOs this presents a significantly more difficult problem as it is less likely there are people available to replace the expert. This of course means it is even more vital that the expert's knowledge is captured. It is these two expenses (direct cost of development and the indirect in-kind cost of losing key personnel for extended periods) that make the development of expert systems prohibitively difficult for SMEs and NPOs.

One approach to reduce the cost issue is to have a system developed by a number of SMEs and/or NPOs in partnership. These enterprises would not necessarily be in the same business but would have a similar problem requiring a solution. Ignoring the obvious issues involved in any partnership of SMEs, such a group would have a significant difficulty in being able to develop a multidiscipline ES. The ES would need to be developed by extracting the knowledge from numerous experts from across all the organisations. For instance, Delisle and St-Pierre observed that "…contrary to large enterprises, SMEs are much more difficult to model and evaluate"[6]. This results from the "…well documented fact that multi-domain, multi-expert knowledge acquisition and modelling constitutes a great challenge" [6].

The third form of ES available to SMEs and NPOs are general solutions to generic problems. These are significantly cheaper but have the restriction that the solutions are general. Most importantly, they are not a system that has captured the SME's expert knowledge. Therefore, while such systems may be useful for a particular problem they are of no use for modelling the enterprises own practises.

Fundamentally, the primary issue preventing SMEs from being in a position to develop personalised ES is the cost for the engineering team and the in-kind support of providing the expert. Therefore, a viable methodology to counter this cost would need to be able to develop the ES with minimal need for a knowledge engineer and without forcing the expert to frequently be out of the work place to have their knowledge extracted. Ideally SMEs would be best served by a system where the expert can provide the knowledge as they perform their normal duties.

3. Ripple-Down Rules (RDR)

Ripple-Down Rules [1] uses a simple exception structure aimed at capturing the context of knowledge. The design of the methodology allows an expert system under development to validate knowledge without the need of a knowledge engineer or expensive testing procedures. In fact this self validation process is so

simple that it completely does away with the need for a knowledge engineer, resulting in a methodology that can be used to develop an ES purely by the expert themselves. The expert simply follows the following steps:

1. Present a case/problem to the system for evaluation
2. Check the solution generated by the ES against a solution proposed by the expert themselves.
3. If the system's conclusion is correct then return to step 1 to present the next case.
4. If the system's conclusion is incorrect then the expert indicates this to the system and tells it what the conclusion should have been.
5. The system will then asks the expert a simple question.
    a. It will provide a list of differences between this case identified by the expert as incorrect and a previous case the expert previously used, referred to as the cornerstone case.
    b. The expert simply selects one or more differences that they believe sufficiently differentiate this new case from the old case.
    c. The system uses the selected attributes to define a new rule for when this new conclusion should be given
6. Return to step 1 for the next case.

That's it! The expert never has to do anything more difficult than use their own knowledge to decide which attributes are the most important in defining the new solution. The beauty of this system is that it *behaves as an expert expects*. The expert tells the computer when it is wrong; tells the computer what the answer should have been; and, gives a justification for the corrected solution.

This simple approach to knowledge acquisition is made possible through RDR's *no-model* approach. RDR uses a binary tree structure of rules. This tree structure actually forces knowledge to be stored in context. When you correct the system it is simply adding a new node at a leaf (bottom) of the tree. Therefore, it is only adding the new rule in the context of the observed error. This also means that the new rule only has to be validated in that localised context, which is why the system works so elegantly. Furthermore, it also means that regardless of the size of the ES, whether this is the 3rd rule or 293rd rule added, the process is always the same.

3.1 Classification

A more technical explanation of the RDR methodology is that each node in the binary tree contains a rule, a conclusion (or classification) and a cornerstone case (section 3.2). Each node has two possible branches, representing whether the rule was satisfied or not by the data provided in the presented case. If a rule is found to be 'true' then the successful branch is followed to the next rule and *vice versa* for the unsatisfied branch [1]. This process continues until either a leaf node (a node with no more branches) is reached or until a node has no appropriate branch to follow. The conclusion returned by RDR is the conclusion from the last successful rule. That is, if the last node tested was true then its conclusion is returned. However, if it was false then its parent-node's conclusion (if its rule was satisfied) is returned [1]. This process of checking the parents is continued until a satisfied rule is found, thereby returning that rule's conclusion.

The root node is a special case, of the form ‘If true then – default conclusion’, which is always satisfied. Therefore, if no other rule is satisfied then the default conclusion is returned, guaranteeing a conclusion will always be found.

For example, a case with the attributes {a, b, c, g, h} is presented to the RDR KB shown in Figure (i). In this tree it can be seen that: rules 1 and 3 have both true and false branches leading to further rules; rule 2 only has a false path; rules 4 and 6 (and obviously the root node) only have a true path; and, rules 5, 7 and 8 are leaf nodes. When the case is presented it ripples down the tree using the path {0 – 1 – 3 – 6} where, because there is no attribute ‘f’ and no false branch, the inferencing process completes. The conclusion returned is 1 from rule 1, due to this being the last rule to be satisfied.

It should be observed in the above example that even though our case had the attributes ‘b’, ‘g’ and ‘h’ and that the RDR tree has rules 2, 5 and 8 that test for these attributes, they were never used, because the context that those attributes are required in never occurred. It should also be noted that the rule numbers have no meaning in the inferencing process and if used in implementation generally indicate the order rules are created. They are shown here only so particular rules can be identified and discussed. Also, it can also be seen that some conclusions are repeated at different locations. This is usually the case as most classifications have multiple possible definitions.

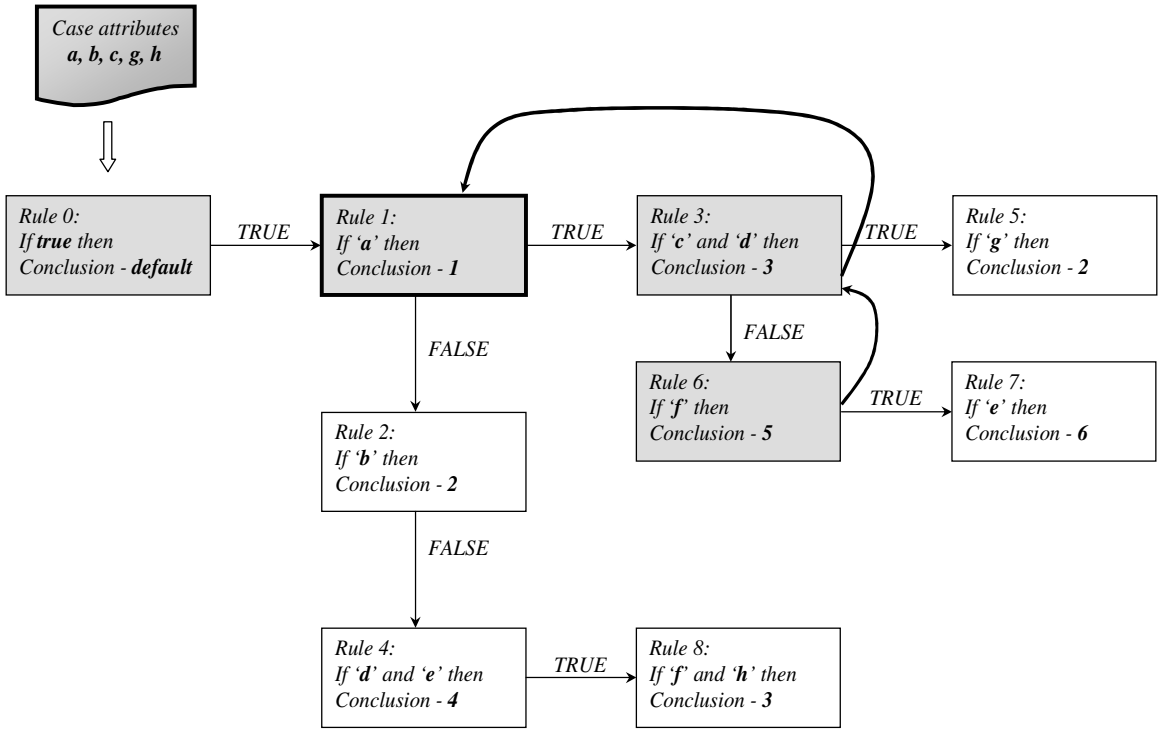

Figure (i): Example of the RDR binary tree structure. The rectangle nodes of the tree contain a rule and a classification. The branches of the tree are either 'true' or 'false'. The straight arrows indicate the direction of inference. The empty block arrow and the document shape indicate a case being presented to the KB for inferencing. The grey filled boxes show the rules that are tested during the inferencing process for the example case. The curved arrows show the direction taken when searching for the last satisfied rule and the bold outline rectangle shows that last satisfied rule. Thus, the final conclusion of the inferencing process is

class ‘1’.

3.2 Learning

As mentioned previously, the primary advantages of RDR are the ease that new knowledge can be added and that the method of adding knowledge is the same, regardless of whether the system is in the development or maintenance stages. The inclusion of new knowledge is simple because of the context based structure of the rules. When a new rule is being added it is merely appended to the tree as a leaf node and it only needs to be validated within this narrow contextual domain.

RDR learns through the acquisition of rules, increasing the size of the KB. If the expert disagrees with the conclusion of the system after inferencing then they simply provide a justification for why it was wrong, which is used as the new rule. The justification is determined by first comparing the current case with the previous case that had originally created the parent rule, referred to as the cornerstone case. A list of differences, in the form of attributes, between these two cases is generated from which the expert selects one or more. Those attributes selected justify the new rule. The new rule created is unable to mistakenly classify the old cornerstone case with the new rule because only differences were used in creating the rule. This concept of using differences is not new and has been used in knowledge acquisition (KA) methodologies based on Kelly’s [7] Personal Construct Psychology (PCP). It is a useful acquisition technique as people are good at identifying differences [8-10]. The RDR approach differs from these other PCP based methods, because, rather than asking the expert to think of differences, RDR simply asks them to select the relevant differences [8].

For example, continuing from the previous example in Figure (i), Figure (ii) illustrates how a new rule is created and added to the RDR structure. It is assumed that the above inference has occurred and the expert has decided the conclusion of class 1 is incorrect. Firstly, the cornerstone case that was used in the creation of rule 6 (this rule is used as it was the last one visited during the inference process) is loaded and compared with our new case identified as having been misclassified. The two cases are merged and the unique attributes extracted, as identified in the difference list. The expert then selects the relevant differences that best distinguish between the documents, for instance ‘h’ and ‘li’ has been selected in this example. The new node is then created by writing a rule with the attributes selected, the correct class given by the expert and our current case. The current case will become the cornerstone case for this new rule ready for future corrections. The new rule is then attached as a child node to rule 6 on the false branch.

As this example shows, through the inclusion of context within the structure of the KB, two major advantages are achieved. Firstly, the cause of the failure by the KBS during inferencing is automatically determined. This significantly improves on existing KA paradigms where a KE will have to study the rule base to locate the cause of an error even before looking at how it is to be corrected. Secondly, when creating the new rule it is guaranteed to be consistent with the existing KB without the need for extensive testing of the new rule [11].

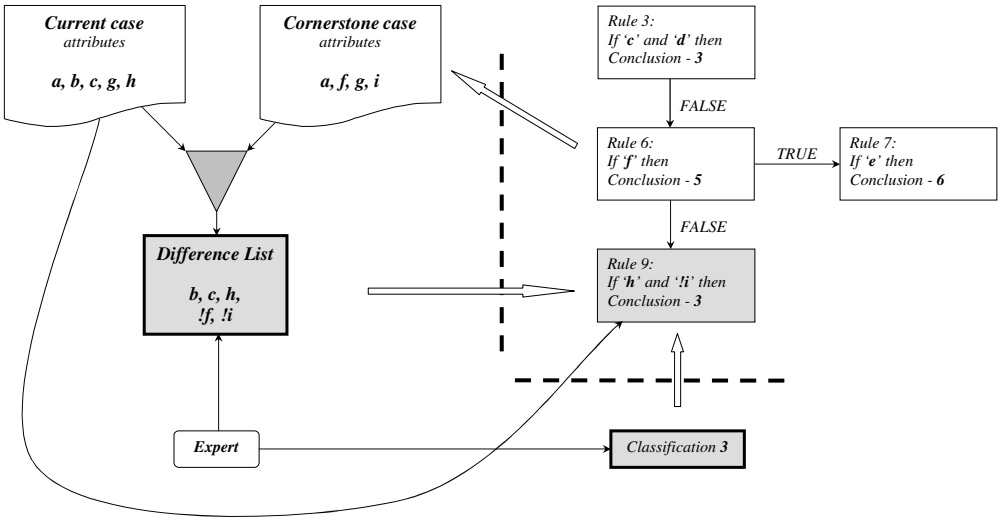

Figure (ii): Example of creating and incorporating new knowledge in RDR. The grey triangle represents the merging of the cornerstone case (extracted from the terminating rule during inferencing) and the current case, producing a list of attributes that differentiate the two cases, shown in the grey box with a bold border. The expert selects from this list the attributes that best identifies the reason for the conclusion. A new leaf node is created incorporating a rule, compiled from the expert's selections, the classification identified by the expert, and the current case which will be the new node's cornerstone case. This node is then added to the tree at the point where the inferencing process was halted.

3.3 Multiple Classification Ripple-Down Rules (MCRDR)

RDR has been shown to be a highly effective tool for knowledge acquisition (KA) and knowledge maintenance (KM) [12, 13]. However, its inability to handle tasks with multiple possible conclusions significantly limits the method's ability to be applied in many domains. To handle this multiple classification issue Multiple Classification Ripple-Down Rules (MCRDR) [12, 13] was developed. For the expert MCRDR operates essentially in much the same way as RDR. The expert presents a case, confirms the classifications given by the application or reclassifies it if the system was wrong. To reclassify the case the expert simply identifies the correct conclusion and answers one *or more* questions to differentiate it from previous cases.

There are two primary differences between RDR and MCRDR in how the expert interacts with the system. The first difference is that they may have to be asked for more than one case comparison. This is due to the multiple possible conclusions but is rarely more than two or three [12]. Secondly, the expert should indicate where in the tree the new rule should be placed. This last step is potentially a problem except a mechanism was developed for the system to automate this process [12, 13]. MCRDR uses an n-ary tree rather than a binary tree, but still stores knowledge within context and is otherwise identical to the original RDR.

4. Rated MCRDR (RM): A Methodology for SMEs and NPOs

In the previous section we described the underlying methodologies this work is built on. However, this material has been available in the literature for some time and has not been taken up by SMEs for developing their own ES. This is because

of one major limitation that was deliberately skipped over in the last section. Recall step 2, *check the solution generated by the ES against what the expert themselves think the solution should be*. The clear flaw in this step is that the *expert must check every case* to see if it is correct. This clearly breaks the whole point of building the expert system in the first place. Recall, the reason for developing an ES is in case the expert becomes unavailable. This problem does not fully negate the use of an RDR methodology as the expert could check every case until the system gets a certain percentage correct (say 95% accuracy). However it is a limiting factor.

Ideally we would want people, other than the expert, to be able to use it during development without needing to also check every case with the expert. For this to be possible we need the application itself to recognise when the classification it gives may be incorrect. Thus, we need the ES to contain meta-knowledge about itself. This post processing of a result could then produce a warning to the non-expert that a particular case should be double checked by the human expert. The ability to perform such post processing is a relatively new field of study commonly referred to as prudence analysis (PA).

Some early systems such as WISE [12, 14], Feature Recognition Prudence (FRP) and Feature Exception Prudence (FEP) [14-16] were only marginally effective, as they produced far too many false positives. A later approach by Compton et al [17] which was improved further by Prayote [18] made a significant step forward, producing much improved results within the single classification problem.

Rated MCRDR (RM) [2] is an adaptation to the MCRDR methodology which can be easily applied to this problem of gathering meta-knowledge about the knowledge base. RM is a hybrid algorithm utilising an artificial neural network [19] to find unknown patterns in a problem's solution. Firstly, a case is presented to the MCRDR tree, which classifies the case. Then for each rule in the inference, an associated input neuron will fire (see figure (iii)). The network then produces a vector of output values, $\bar{v}$, for the case presented.

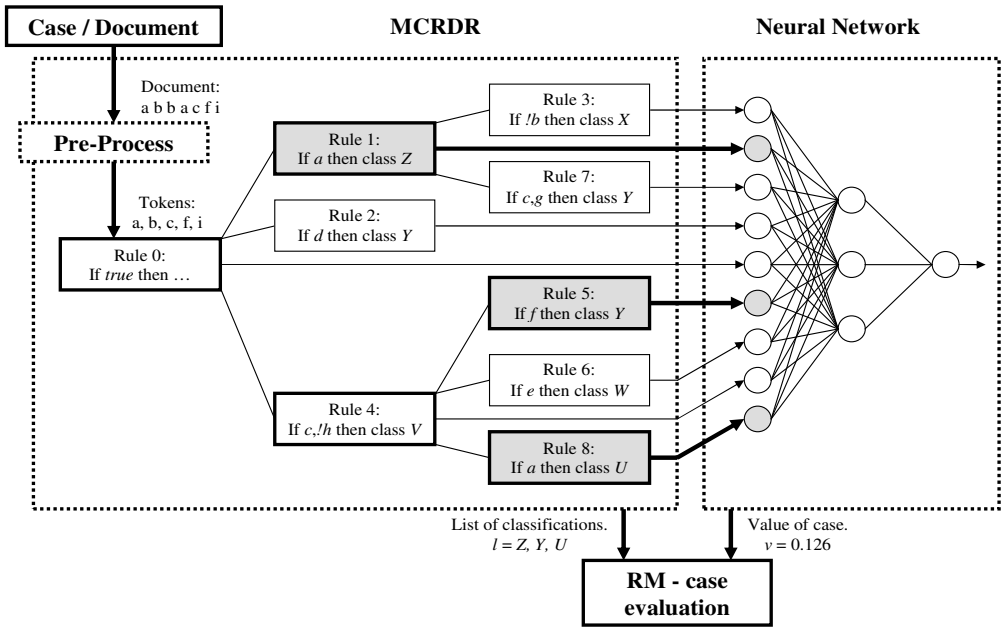

Figure (iii) RM illustrated diagrammatically

The basic idea behind applying RM to PA is to allow MCRDR to develop classifications in the general way, while the network passively watches rules being added to the MCRDR tree. As it watches it also attempts to identify the correct classifications. Through classification testing it has been found that the classifications between the MCRDR component and the network often differed when MCRDR misclassified [2]. Therefore, the prudence system developed identifies these differences and warns the user that the classification by MCRDR could be wrong.

Training is a simple process of identifying the correct classification that the expert has agreed to when accepting a case. Obviously, however, this can only be done when a warning has actually been generated. When no warning is generated the system is unable to train because the system cannot be certain whether the expert would have wanted to alter the classification. When a warning is given and the expert confirms a classification the reward is a positive value at the output where it should have been classified as a particular case and a negative value otherwise.

RM has been tested on the same datasets as [17]'s earlier work and results indicate that the system produces a greater degree of accuracy with a similar number of warnings (table (i)), especially on the larger and more complex dataset - GARVAN. However, it was found during testing that through the adjustment of a single variable, RM could improve accuracy at the expense of producing more warnings or produce fewer warnings with a lower accuracy. This result is illustrated in figure (iv).

Effectively, this simple adjustment allows an expert direct and simple control over the warning and accuracy of the prudence system and therefore the resulting ES. It can be seen in these results that RM was able to achieve virtually 100% accuracy. For example, the system can achieve nearly 100% accuracy (the actual average result was 99.93%) if warnings are provided on just over 50% of cases. This is clearly a lot of warnings but is a vast improvement on the current

use of RDR methodologies where the expert must check every case. What is important in these results is that you can achieve a result very close to perfect with far fewer warnings. The other aspect of these results is that the number of warnings can be reduced to around 84%, in the GARVAN dataset, of cases if the expert can tolerate an accuracy of just fewer than 90%, which is still very high.

Datasets	Algorithms	False Neg %	True Pos %	False Pos %	True Neg %	Accuracy %
GARVAN	*RM*	0.1	1.7	15	83	92.9
	Compton	0.2	2.4	15	83	91.7
Chess	*RM*	0.2	0.8	8	91	80.0
	Compton	0.3	1.3	7	91	81.3
Tic-Tac-Toe	*RM*	2.9	8.8	12	76	75.2
	Compton	1.5	3.8	14	81	71.7

Table (i): Comparison of the averages between the RM and [17]'s work (labelled Compton). These results have been rounded to 2 significant figures. The four columns of raw results are shown plus the calculated accuracy.

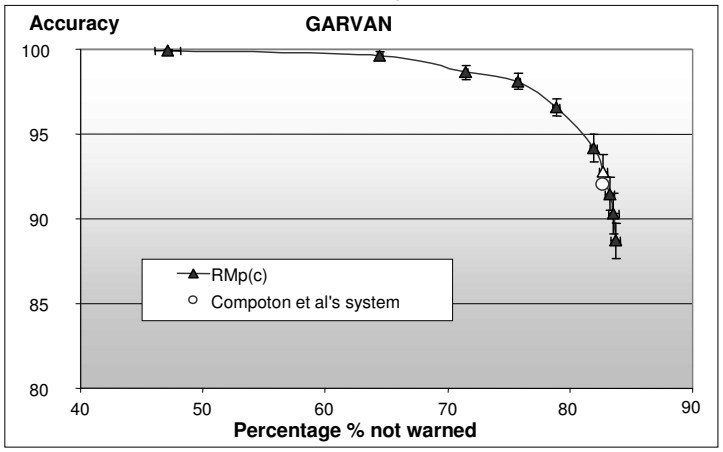

Fig. (iv). Compares the results using different threshold adjustment values. The y-axis represents the level of accuracy achieved by each experiment. The x-axis shows the percentage of cases where a warning was correctly not given.

The benefit of a system such as this in the development of expert systems would be most noticeable when developing for an SME or NPO. This system allows the expert to set the number of warnings, and therefore, the level of accuracy, potentially through the use of a simple slide bar. Furthermore, they no longer are required to check every situation given to the ES even during development. Therefore, this methodology provides a means of developing an expert system without the need of an expensive knowledge engineer or the time commitments from the expert required by traditional expert system development methodologies.

6. Conclusion

This paper has discussed some of the issues for Small to Medium sized Enterprises (SME) and Non-profit Organisations (NPO) implementing expert systems (ES) technology for the capture of their unique business knowledge. In this discussion it was claimed that current ES development methodologies rely heavily on the use of knowledge engineers which are very expensive. The in-kind

cost to a business of their expert providing extensive time to the development of an ES was also prohibitively expensive and disruptive to the organisation.

This paper then went on to introduce a new ES development methodology that uses a *no-model* approach to knowledge acquisition, and therefore, does not require a knowledge engineer. This methodology also provides a prudence analysis system that allows the expert to avoid supervising the system underdevelopment all the time. Instead, the expert only needs to add rules when a warning of a pending misclassification is produced. Results were provided that show the ability of the prudence system to predict gaps in the domain knowledge. This methodology allows the development of expert systems cheaply and easily making this a viable approach for SMEs and NPOs.

References

[1] Compton, P., and Jansen, R., (1988) Knowledge in Context: a strategy for expert system maintenance, *Second Australian Joint Artificial Intelligence Conference (AI88),* Vol. 1, pp. 292-306.

[2] Dazeley, R., (2007) To The Knowledge Frontier and Beyond: A Hybrid System for Incremental Contextual-Learning and Prudence Analysis, pp. 281. University of Tasmania (UTas) Hobart

[3] Newell, A., and Simon, H.A., (1976) Computer Science as empirical Inquiry: Symbols and Search, *Communications of the ACM,* Vol. 19, No. 3, pp. 113-126.

[4] Wielinga, B., Schreiber, G., and Breuker, J., (1992) KADS: a modelling approach to knowledge engineering, *Knowledge Acquisition,* Vol. 4, pp. 5-53.

[5] Compton, P., Kang, B.H., Preston, P., (1993) Knowledge Acquisition Without Analysis, *Knowledge Acquisition for Knowledge Based Systems,* Vol. 723, Springer Verlag, Berlin, pp. 278-299.

[6] Delisle, S., and St-Pierre, J., (2004) Expertise in a hybrid diagnostic-recommendation system for SMEs: a successful real-life application, *Proceedings of the 17th international conference on Innovations in applied artificial intelligence,* Vol. 3029, Springer Verlag Inc, pp. 807 - 816.

[7] Kelly, G.A., (1955) The Psychology of Personal Constructs, Norton, New York,

[8] Compton, P., Edwards, G., Kang, B., (1991) Ripple Down Rules: Possibilities and Limitations, *6th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW91),* Vol. 1, SRDG publications, Canada, pp. 6.1-6.18.

[9] Gaines, B., and Shaw, M.L.G., (1990) Cognitive and Logical Foundations of Knowledge Acquisition, *5th AAAI Knowledge Acquisition for Knowledge Based Systems Workshop,* Banff, Canada,

[10] Shaw, M.L.G., and Gaines, B., (1992) Kelly's "Geometry of Psychological Space" and its significance for Cognitive Modeling, *The New Psychologist,* pp. 23-31.

[11] Beydoun, G., (2000) Incremental Acquisition of Search Control Heuristics (PhD thesis), University of New South Wales (UNSW) Sydney.

[12] Kang, B., (1996) Validating Knowledge Acquisition: Multiple Classification Ripple Down Rules (PhD thesis), University of New South Wales (UNSW) Sydney.

[13] Kang, B.H., Compton, P., and Preston, P., (1995) Multiple Classification Ripple Down Rules: Evaluation and Possibilities, *The 9th Knowledge Acquisition for Knowledge Based Systems Workshop,* SRDG Publications, Department of Computer Science, University of Calgary, Banff, Canada,

[14] Edwards, G., (1996) Reflective Expert Systems in Clinical Pathology (MD thesis), University of New South Wales (UNSW) Sydney.

[15] Edwards, G., Kang, B.H., Preston, P., (1995) Prudent expert systems with credentials: managing the expertise of decision support systems, *International Journal of Bio-Medical Computing,* Vol. 40, pp. 125-132.

[16] Edwards, G., Compton, P., Kang, B., (1995) New Paradigms for Expert Systems in Healthcare, *Proceedings of the Fourth Health Informatics of NSW Conference,* Primbee, pp. 67-72.

[17] Compton, P., and Preston, P., (1996) Knowledge based systems that have some idea of their limits, *10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW99),* SRDG publications, Canada.

[18] Prayote, A., (2007) Knowledge Based Anomaly Detection (PhD thesis) University of New South Wales (UNSW) Sydney.

[19] Beale, R. and Jackson, T. (1990). Neural Computing: An Introduction. Bristol, Great Britain, IOP Publishing Ltd.