Closing the Gap Between Different Knowledge Sources and Types in the Call Centre

Debbie Richards and Megan Vazey

Department of Computing,
Division of Information and Communication Sciences,
Macquarie University,
Sydney, NSW 2109, Australia.
richards@ics.mq.edu.au, mvazey@ics.mq.edu.au

Abstract

Our current project involves improving the trouble-shooting process in the support centre of a large multinational organisation in the Information and Communications Technology (ICT) industry. What has become obvious is the need to capture and reuse many different types of knowledge from a wide range of sources. We have conducted an evaluation study within the organisation to identify the types and sources of knowledge used, the rate of repeat problems and solutions and what improvements are needed. We provide some of our results in this paper. We present an approach known as FastFIX that supports the acquisition and reuse of troubleshooting knowledge from multiple sources using links to relevant intranet and internet-based material. Our system seeks to align the goals of the support-centre, such as maintainability and workflow compatibility, and can inter-operate with the support-centre's existing problem ticketing and knowledge management systems.

KEYWORDS

Knowledge based systems, knowledge acquisition, Ripple Down Rules, RDR, Multiple Classification Ripple Down Rules, MCRDR, Support Centre, Call Centre, Help Desk, Service Desk.

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, libraries and even databases, to search engines and hyperlinks. Despite the information revolution heralded by client-server internet technology, the problem for the support-centre help-desk / service-desk remains the same: how can we achieve rapid access to the minimum set of knowledge required to solve the problem on hand?

Software solutions for the call-centre are typically focused on providing defect tracking by recording: who raised the call; what product is involved; what are the environment variables such as the operating system; to whom the problem is assigned; and the current status. We are not simply focused on keeping track of the problem. We want to assist with finding a solution based on what solutions have been used in the past and the symptoms that make this solution applicable. More recently vendors have offered software that documents solutions turning the troubleshooting task into one of searching and matching a database of known solutions. In the organisation that we are studying, two separate databases are maintained, one containing problems and the other containing solutions. What is missing is the link between the two. Call centre operators need to use problem determination knowledge, which is problem solving knowledge that allows an expert to determine the class of problem on hand. In addition, and what is typically missing, is search knowledge, which is the where-to-search and what-to-search-for knowledge that allows an expert to find an appropriate solution. A key differentiating feature of our solution over many current vendor solutions is that it offers a closed-loop feedback system whereby users continually update and refine system search results, and hence the system knowledge as part of their daily work effort. We have significantly extended the Multiple Classification Ripple Down Rules (MCRDR) technology, initially introduced by Kang, Compton and Preston (1995) to handle the complexity of the support-centre environment in the ICT as well as other support domains, as a result of technology convergence, vendor divergence, product evolution, staff mobility, and multiple and potentially conflicting views on how to solve a wide range of problems.

Firstly, we provide a brief review of some related work. We then present the problem in more detail using the results from surveys, interviews and observations conducted in the organisation. We then provide an introduction to MCRDR technology, our design goals and our system architecture. Finally, we present our conclusions and next steps.

RELATED WORK

Technological solutions to the Help Desk / Call Centre / Support Centre have been offered by countless software vendors, many of them offering some degree of intelligence. In 2004, the website http://www.helpdesk.com listed 314 vendors of "Help Desk" software, 26 vendors of "Knowledge Management" software, 133 vendors of "CRM and Call Centre" software, and 7 vendors of "Defect Tracking" software. Similarly, http://www.helpdesksoftware.org (2004) had an extensive list of software vendors providing knowledge management software for the helpdesk. While we have not performed an extensive study of these products, due to a lack of sufficiently detailed product descriptions, we found that many products at most provided sophisticated problem / document / case management, but did not provide intelligence beyond limited inconsistency checking, assistance with template filling or simple problem / document / case searching.

The Information and Communication Technologies (ICT) domain is notorious for being jargon-rich. Some of the reasons why current popular search approaches (such as google) may not work well on their own when searching for solutions in this domain are as follows:

- Simple keyword searching may not be sufficient to locate the relevant information. A richer grep/sed style pattern matching mechanism is needed, for example to match sequences of hexadecimal error codes, or patterns of version number, and at times using strategically placed wildcards.
- 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 search will be greatly enriched (i.e. fewer false positives returned) by constraining search queries to within fields provided by the existing data structures.
- Custom ontologies are often required to define organisation and domain specific jargon, for example that 'version' is the same as 'release' or 'edition' or 'issue' or 'edition' but not the same as 'iteration'.
- Current information is needed as real-time data, as opposed to cached, and therefore non-current data.

Following on from the initial work of Roger Shank in the early 80s, numerous case-based reasoning (CBR) systems have been developed for the help desk domain including: SMART (Acorn, 1992), CASCADE (Simoudis, 1992) and CARET (Shimazu, et al., 1994). However, Kim et al. (1999) notes that the methods used by CBR systems to index, compare and modify cases, necessitate a degree of knowledge engineering expertise. Any knowledge engineering effort can easily become a bottleneck in such systems. We think that traditional rule based and case based solutions fall short of the target for the following key reasons:

- Firstly, knowledge exists more as a relational network structure, rather than a top-down hierarchical tree structure.
- Secondly, human capacity to comprehend either a top-down rule tree structure, or a network knowledge structure is limited. Our *rationality* is unfortunately *bounded*.
- Thirdly, a key observation is that troubleshooting knowledge is learned, acquired, generated and consumed in a decentralized manner across the global support organization. Centralising the control of such knowledge may create greater consistency of expression, but may come at the expense of responsiveness by the support organization. An *elastic organisation* will need to manage the decentralized capture and re-use of knowledge, instead of, or perhaps in parallel with, a centralized knowledge engineering function.

Tsoukas and Vladimirou (2001) reviewed the flow of organisational knowledge within a customer care call centre for Panafon, Greece's leading mobile phone operator in 2001. They observed that despite the Panafon call centre not being a knowledge-intensive environment, and despite the employee perception that answers to 95% of the questions asked were available "somewhere" in the computer system, several operators were observed constructing their own personal information systems, which contained photocopies of the relevant corporate manuals plus personal notes. In other words, alongside the formal organisational knowledge there existed an informal knowledge that was generated in action, and which represented the heuristic knowledge residing both in individual's minds and in the stories shared in their communities of practice.

Adria and Chowdhury (2002) believe that the ultimate purpose or effect of a call-centre implementation "is to streamline the pathway to information for the customer". They identified three dimensions of call centre employee skill: *responsibility*, *abstractness* and *interdependence*. They used these three dimensions to argue that call centres should allow decisions to be made as close as possible to the customer, including the employee decisions to add to, revise and work with the corporation's knowledge base. They identified the case of Sun Life, a group insurance company, that took a team approach involving both front-line workers and technical experts to designing the service delivery operations so that the customer could experience a richer real-time interaction. They also highlighted the case of the Mayo Clinic in Rochester, Minnesota, where physicians post links on the clinic's intranet to web sites that provide up-to-date and authoritative information about current medical treatments for use by clinic practitioners. For call centres to thrive, they argue that agents need an adequate amount of autonomy and responsibility, and that agents can have a role in updating and correcting the knowledge base.

Our approach is a practical application of such ideas in the call centre / support centre or help desk as it compels agents to collaboratively build up a knowledge base, and a set of organising rules for that knowledge, using real cases that they deal with on a daily basis.

CLARIFYING THE PROBLEM

Capturing knowledge is useful when it is difficult to discover or retrieve that knowledge in the first place, and that knowledge is needed time and time again. If it is time consuming to find a solution and that solution can be reapplied to reduce the resolution time then the organisation can better manage its knowledge sources by shortening this solution search time. Such efforts may provide a number of benefits such as improved productivity, shortened response times to customers, and reduced worker frustration. Thus, we were interested in determining if the same problem is seen multiple times and the time taken each time to solve that problem. As can be seen in Figure 1, we identified a number of problems (labelled A-G) faced by the call centre support staff and recorded how many problem report incidents they appeared in and how many hours (y-axis) it took to resolve. A number of possibilities became apparent. For instance, Problem C, took about 30 hours to solve, but once it was solved in the first incidence, 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 this data is influenced by both staff and client availability, on the whole the data indicates that the knowledge was lost and had to be rediscovered, repeatedly.

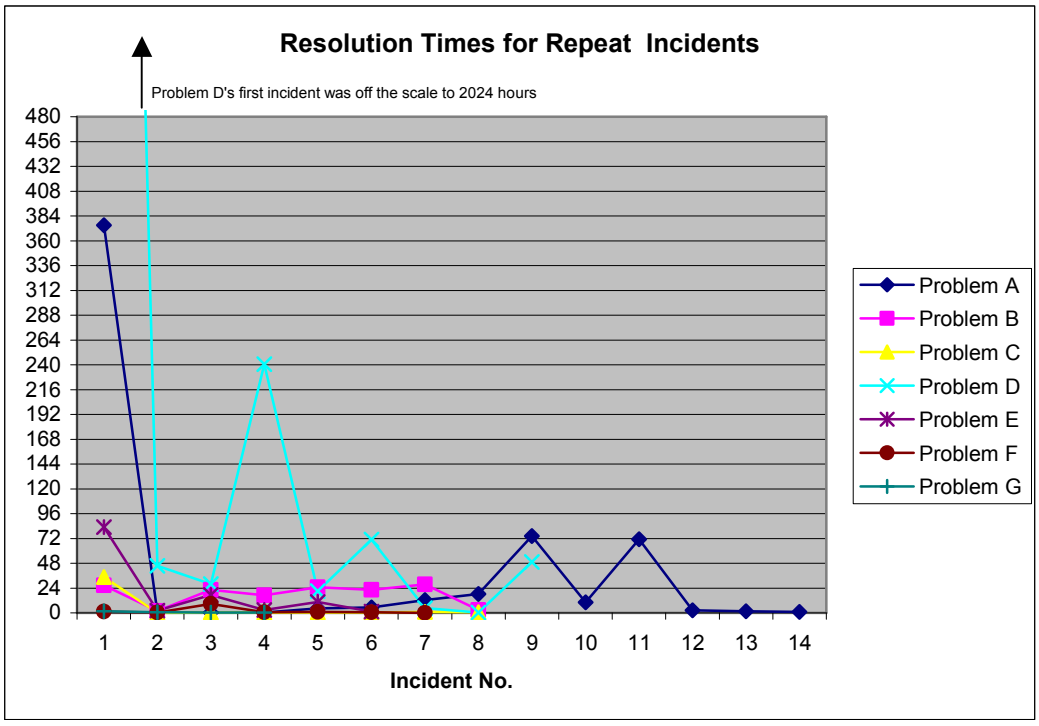

Figure 1: Time taken in hours (Y-axis) to solve the same problem seen in multiple incidents (X-axis).

To confirm what this initial data indicated, we conducted a number of surveys within our organisation. The surveys had the following objectives:

1. To determine the current 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 troubleshooting process.

The survey contained two parts: survey Part A which included 67 questions and took participants on average 50 minutes to complete and survey Part B which included 11 questions and took participants on average 5 minutes to complete. The products supported by the call centre are divided into product ranges and these are assigned to different knowledge workers. There are 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. The study covered support staff from both support levels and two different product ranges, resulting in 4 groups of approximately 5 staff and hence 20 staff in total.

We provide a snippet of the results to the 78 questions. Respondents believed that: 76% of customer problems assigned to them would be seen again by themselves or others within the organisation; and 64% of customer problems assigned to them had been previously seen by themselves or others i.e. they were repeat problems. From this we can deduce that around 12% of problems will be "first-time" problems that will reoccur.

Further, respondents believed that: in 67% of cases they would not know the solution straight away and would need to refer to other sources of information to solve the case; and in 43% of cases, they would involve their team-mates or others in solving the problem. Table 1 gives a breakdown of the resources used for problem solving. For *at least* 48% of the mundane routine repetitive types of problems that have been seen before, and that will be seen again, respondents indicated that they would need to refer to other sources of information to solve the problem! These mundane problems comprise 64% of the case load!

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

Table 1: Most to least frequently relied upon resources

In answer to "What are the biggest roadblocks that stop you being effective and efficient in your role?" we received:

- Training / Knowledge (8 responses);
- Accessible documentation (6 responses);
- Solution database (3 responses);
- Time pressures (3 responses);
- Escalations (2 responses);
- Customer contact (2 responses).

When asked "Have you had all the training that you require to perform your troubleshooting role effectively and efficiently?" 18% said yes, 23% were undecided and 59% said no. This prompted us to ask why training was inadequate despite the organisation's extensive training program. The answer appeared to be that solutions required global sources and audiences and involved a range of knowledge types.

Based on interviews and observation, we identified the following types of knowledge:

- 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 can you determine what the problem is? How can you fix it?

Further, we found that:

- much of the knowledge was stored in people's heads (i.e. tacit) rather than documented in technical references (i.e. explicit). Existing documentation often missed the necessary detail or was ambiguous. Product information was cryptic at best, coming in the form of abbreviated slides, videos, or emails.
- Personal relationships were extensively relied upon to extract basic product knowledge from scattered sources across the company.

The impact of these fragmented knowledge networks is a poor level of knowledge re-use that results in:

- increased frustration,
- duplication of effort,
- slower problem resolution,
- inconsistent customer responses,
- customer dissatisfaction, and
- overall organisational inefficiency.

We note that high staff turnover rates are both an outcome and a contributing factor.

RIPPLE DOWN RULES

We offer an approach that supports a Community of Practice by allowing knowledge workers in the call centre to collaboratively build a knowledge base of solutions motivated by the problem incidents as they arrive. Our solution is a hybrid case-based and rule-based approach that involves a simple incremental technique for knowledge acquisition. The approach is built on a philosophy, known as Ripple Down Rules (Compton and Jansen 1989) where knowledge is seen to be socially situated, contextual and continually changing and emerging. Another noteworthy feature of the philosophy is that it acquires both codified and tacit knowledge as it captures knowledge in action as domain experts interact with cases.

We have chosen Ripple Down Rules over other knowledge acquisition techniques because, to the best of our knowledge, it is the only technique that supports the incremental capture of a sequence of classification rules in the specific context of the data being classified.

Ripple Down Rules is a technique for building and maintaining knowledge based systems that facilitates the natural human process of knowledge acquisition. The technique was conceived as a result of a realisation that 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. This realisation is critical since for each case presented to an expert, the expert provides and justifies the conclusion in a particular and specific context. The conclusion may not be appropriate for other contexts that the expert did not consider.

In our extension of Multiple Classification Ripple Down Rules (MCRDR) (Kang, Compton and Preston, 1995), we observe that the normal process of knowledge acquisition involves continuous refinement of the expert justifications that attempt to explain why a particular case belongs to a given classification and therefore deserves a given set of conclusions.

MCRDR builds up a decision tree that allows multiple classifications to be fetched when cases are evaluated against the tree. Working from top to bottom through the tree, when a RuleNode in the decision tree evaluates to `TRUE` for a case, the case is then evaluated against all its child RuleNodes. The classifications are given by the last `TRUE` RuleNode in each and every path of the tree. When a classification is given that the expert disagrees with, the expert is asked to identify features of this case that distinguish it from cases classified by the parent RuleNode, and to formulate a rule based on those features that is used to construct the child RuleNode. The net result is an incrementally built decision tree that can be used to classify cases in any given domain.

We have extended this knowledge acquisition and representation technique to develop an expert system that can operate independently and that can be used to augment any legacy case tracking ticketing and knowledge based systems that the organization may have already invested in. MCRDR has found commercial success, particularly in the pathology domain (Lazarus 2000) as maintenance of conclusions and rules is a simple task performed by the domain expert. However, the MCRDR algorithm and representation currently does not support changes to previously seen cases or allow multiple domain experts to update the knowledge simultaneously. The knowledge acquisition technique also needs to be adapted to fit with the complex workflow in the call centre environment.

Our system records the decision intelligence that support / call centre and help / service desk personnel use to determine the resources that best assist with particular types of customer query.

DESIGN GOALS

Expediency, accuracy, and efficiency are key performance criteria for the call-centre. This means that our solution needs to be designed for minimal user-decisions. One problem presented by existing defect tracking and knowledge management solutions is that users are presented with long lists of Attribute-Value (A-V) pairs, many of which are irrelevant to the problem on-hand. It is left to the user to apply a mental filter for each new case when filling in these A-V pairs.

Importantly, we aim to reduce the decision burden for users of the system, and thereby speed up the process of problem determination as well as reduce the risk of information overload to the user. In this endeavor, we apply and extend two variations to the RDR theme: Recursive RDR (R-RDR) (Mulholland et al 1993) that involved repeated inference cycles using the single classification RDR structure; and Interactive RDR (I-RDR) which was a technique that allowed the RDR system to prompt the user for more information when required. Hence we apply IR-MCRDR.

Our idea is to use IR-MCRDR in a configuration sense. Our system assists the user in honing the problem definition by using the current case context to prompt the user for more detail about the specific problem being considered. Only relevant A-V entries will be requested of the user, depending on the current case context.

In addition, as more A-V pairs are gathered to define the case, the case will invoke conclusions that lie deeper down the rule paths of the decision tree, and the conclusions displayed will become more specific to the particular problem being observed. Our hope is that this approach will quickly guide users to the most relevant conclusions for the problem case under consideration.

If one can imagine the famous Microsoft Paper-Clip assistant, our tool will provide a similar question-answer interface to users. The difference is that our users can collaboratively create both the questions and the answers; as well as use the information provided therein; and rate its suitability to the task at hand.

Space does not allow detailed description of and our rationale for the extensive modifications we have implemented in our prototype FastFIX system. However, we provide the following brief description of the key ideas.

Relaxing the Case Differentiation Test for new RuleNodes: With previous implementations of MCRDR, a new RuleNode can only be added when the new rule differentiates the case in question from all the cornerstone cases at the node above. We propose that the user optionally differentiate against a limited number of cases at the parent node, namely those that present a unique set of A-V pairs, and possibly those that meet a certain expiration threshold.

Separation between live and registered classifications: We introduce the notion of *live* classifications which the system continuously calculates for every case in the system, and *registered* classifications that users have previously confirmed as being true for a given case.

Managing approvals: We provide a mechanism that allows one or more trusted experts to indicate their approval of a given classification and its conclusions.

Recording a Change History per Case and per RuleNode: Given that old and expired cases may *fall through* to new conclusions, or that old cases or even old classifications may be edited and modified, a change history can be kept per case showing how the classification list has evolved over time. Similarly, given that classifications containing web references may expire and require update, a change history per RuleNode (that, amongst other things, records the change history of classifications at that RuleNode) is also required.

Building credibility: We intend to satisfy user-demand for credibility by keeping a record at each RuleNode of how many cases presently refer to that RuleNode for classifications that are both *live* and *registered*, together with a path trace for each classification. We see this as fulfilling the role of an explanation of worthiness for system generated conclusions. As discussed by Doyle, Tsymbal, and Cunningham (2003) the use of explanations increases user acceptance of the predictions offered by knowledge based systems.

Optimising the rule tree to handle large numbers of cases: This essentially involves a strategy for minimizing the frequency of evaluation of rule nodes.

Feedback and collaboration: In the true collaborative[1] style of the internet, feedback will be solicited from users to rate the quality of knowledge presented in the context of the given case. There are many reference-rich knowledge bases[2] on the internet. Our aim is use a similar presentation style to capture and present pertinent troubleshooting information for our users.

Enriching the search: Some of the case-based reasoning (nearest neighbour algorithm), formal concept analysis (concept lattice) or data-mining techniques used by others could work in concert with our FastFIX solution to provide useful knowledge paths that will enrich the user experience, particularly when the conclusions presented are insufficient for them to resolve the problem at hand.

Managing incentives: As mentioned earlier, it is vital to the success of our tool that it actually gets used. To encourage maintenance and usage of the system we intend to keep a record of user click-through activity[3] – how many cases they have worked on, how many conclusions they have created, how many conclusions they have *registered* (or *approved* in the case of expert users), the average user rating for conclusions that they have created, and so on.

Conclusion Checking: We intend to limit the scope of conclusions to one or more web URLs. In addition, we intend to explore the option of allowing several conclusions to be recorded at the same RuleNode. Since URLs frequently expire, a test is required to check the currency of URLs presented, and an opportunity to globally edit and update the web references (read conclusions) must be provided.

Accessibility: The need for accessibility has encouraged us to design our client-server solution with a web-browser front-end. If the system is extended beyond the local service-desk organization to assist the global service-desk, separate language interfaces (e.g. Cantonese, Spanish, Japanese) may be required.

Usability: Usability is a key ingredient to success of such a system since the more the system gets used, the quicker the knowledge base will mature and the more compelling the content will become. Usability extends to system responsiveness, ergonomics, ease-of-use and the intuitiveness of the interface.

OUR FASTFIX SYSTEM

At the support centre, 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.

In our FastFIX solution, as a new case 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 – 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.

In our system, Cases 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); possibly 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. Cases may contain a history of case statements that have been added to the case over time by users.

ConditionNodes (i.e. RuleNodes) contain a Rule Statement, which is a boolean expression that may include pattern matching, comparative, or custom operators that can be evaluated by the FastFIX engine to determine the truth of that rule / condition for a given case.

Conclusions are statements which can include internet URLs, plain text, or instructions to the FastFIX engine to interactively prompt the user for more A-V details and then recursively re-evaluate the case. One typical approach would have conclusions that are sets of intranet URLs pointing to existing solutions in a corporation knowledge-base or document management system.

---

[1] Some great examples of collaboration on the internet can be found at: http://en.wikipedia.org/wiki/Main_Page, http://www.codeproject.com/, http://au2.php.net/manual/en/ http://www.insidepolitics.com.au/cgi-bin/Ultimatebb.cgi

[2] Some great examples of reference-rich knowledge bases can be found at: http://portal.acm.org/, http://taylorandfrancis.metapress.com/app/home, http://citeseer.ist.psu.edu/ http://www.springerlink.com/app/home/

[3] Some great examples of click-through and feedback-driven computing on the internet can be found at http://www.amazon.com, http://www.ebay.com.au, http://www.google.com.au

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 ConditionNodes that they have created. Together with an indication of how long they've been using the system, these counts are used to assign users with an overall credibility score. The intrinsic motivation of users to increase their User Credibility Score may be augmented with extrinsic individual and / or team motivators. As well, credibility scores are kept for each entity in the knowledge base itself.

The FastFIX solution offers a very lightweight information broker that acts as an index to knowledge resources across the organisation's intranet and across the broader internet.

Figure 4 shows the top-level architecture of our prototype FastFIX system. We are in the process of evaluating the usability and performance of this system architecture and design.

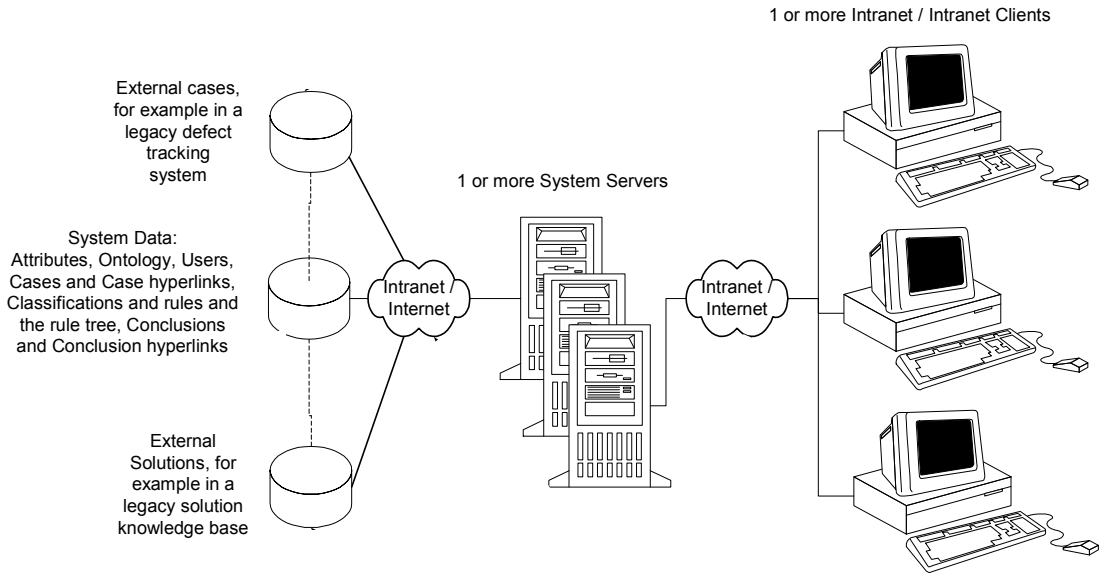

Figure 4: The FastFIX 5Cs System

Our design provides for the possibility that Attributes, Ontologies, Cases, Conditions, Classifications, and Conclusions may be the subject of Collaborative 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.

CONCLUSION AND NEXT STEPS

Imagine, when a new problem ticket comes in, the customer service personnel is presented with a set of tailored refinement queries that enable 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* is presented to the user – which internet links proved useful, and which knowledge-base references helped.

As time goes on, the cumulative effect of presenting more and more cases to the system is that it gets trained to achieve high levels of accuracy in matching solution resources to problem types. This will obviously be of huge benefit to the helpdesk - no more fumbling around with search engines, local web pages, or existing knowledge bases to find the relevant information. Our survey of a global IT support centre indicated that agents spent on average 90 minutes per day just searching for information. Experience with the MCRDR algorithm elsewhere (Kang et al. 1996) suggests that such an expert system will grow rapidly in its level of matching accuracy as cases are added.

In addition, self-maintenance is central to the design of the MCRDR system such that when completely new problem domains are added, the system immediately starts training itself towards coverage of the new domain.

In fact, the system can be configured to identify areas where knowledge is lacking and it can prompt users accordingly. Our strategy keeps open the possibility of data-mining existing knowledge bases, for example to extract previously recorded decision data that may have linked incoming problem tickets to formally constructed solutions.

Our prototype system uses hyperlinks to inter-work with existing problem ticketing and knowledge base systems. Through the trial of our prototype and our investigations we will 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 the ongoing learning of the expert system.

REFERENCES

Adria & Chowdhury (2002) Making room for the call center, *Information Systems Management*, 1-80.

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.

Compton, P. J. and R. Jansen (1989). A philosophical basis for knowledge acquisition. 3rd European Knowledge Acquisition for Knowledge-Based Systems Workshop, Paris: 75-89.

Doyle, D., Tsymbal, A. and Cunningham, P. (2003). A Review of Explanation and Explanation in Case-Based Reasoning. D2003, TCD CS report http://www.cs.tcd.ie/publications/tech-reports/reports.03/TCD-CS-2003-41.pdf]

Kang, B., P. Compton and P. Preston (1995). Multiple Classification Ripple Down Rules : Evaluation and Possibilities. Proceedings of the 9th AAAI-Sponsored Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada, University of Calgary.

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: Pacific Rim Knowledge Acquisition Workshop, 1996 (Sydney, Dept. of Artificial Intelligence, School of Computer Science and Engineering, UNSW, 1996), pp 313-332.

Kim, M., Compton, P., Kang, B. (1999) Incremental development of a web-based help desk system. Proceedings of the 4th Australian Knowledge Acquisition Workshop (AKAW 99), University of NSW, Sydney, 5-6 Dec 1999, 13-29.

Kim M. (2003) PhD thesis, Document Management and Retrieval for Specialised Domains: An Evolutionary User-Based Approach, UNSW, 2003.

Lazarus, L. (2000) Clinical Decision Support Systems: Background and Role in Clinical Support, http://www.pks.com.au/CDSS_White_Paper_doc.pdf

Mulholland, M., Preston, P., Sammut, C., Hilbert, B. and Compton, P. (1993) An Expert System for Ion Chromatography Developed using Machine Learning and Knowledge in Context *Proc. of the 6th Int. Conf. On Industrial and Engineering Applications of Artificial Intelligence and Expert Systems* Edinburgh.

Simoudis, E. (1992) Using case-based reasoning for customer technical support *IEEE Expert* 7(5): 7-13.

Shimazu, H., Shibata, A. and Nihei, K. (1994) Case-Based Retrieval Interface Adapted to Customer-Initiated Dialogues in Help Desk Operations. AAAI 1994: 513-518

Tsoukas H., Vladimirou E. (2001). What is Organizational Knowledge? *Journal of Management Studies*, Vol. 38 Issue 7 Page 973 November 2001.

ACKNOWLEDGEMENTS

Many thanks to our industry partner for their ongoing support in this project. This project was funded by an ARC Linkage Grant.

COPYRIGHT