ICARUS : Design and Deployment of a Case-Based Reasoning System for Locomotive Diagnostics

Anil Varma[1]

[1]Information Technology Laboratory,
General Electric Corporate Research and Development,
Niskayuna, NY 12301
varma@crd.ge.com

ICARUS : Design and Deployment of a Case-Based Reasoning System for Locomotive Diagnostics

Anil Varma[1]

[1]Information Technology Laboratory,
General Electric Corporate Research and Development,
Niskayuna, NY 12301
varma@crd.ge.com

Keywords : Case-based reasoning, locomotive, diagnosis, feature extraction, feature weighing, fault codes

Abstract. Locomotives, like many modern complex machines, are equipped with the capability to generate on-board fault messages indicating the presence of anomalous conditions. Such messages tend to generate in large quantities and difficult and time consuming to interpret manually. This paper presents the design and development of a case-based reasoning system for diagnosing locomotive faults using such fault messages as input. The process of using historical repair data and expert input for case generation and validation is described. An algorithm for case matching is presented along with some results on pilot data.

1 Introduction

There is a recent move in industry towards supporting equipment servicing as a means of augmenting traditional revenue sources such as those generated by equipment sales with limited warrenties and subsequent parts supply. This is especially applicable in the case of heavy machinery which due to its design complexity is often best serviced by the manufacturer. Examples in case include gas turbines, aircraft engines and locomotives. With the emergence of long-term service contracts for such equipment, it is essential that the manufacture minimize its cost of service by proactive on-board and off-board monitoring and diagnosis. Within this context, this paper describes the development of ICARUS (Intelligent Case-based Analysis for Railroad Uptime Support) - a case-based reasoning tool for off-board locomotive diagnosis for use by GE Transportation Systems.

Locomotives are complex electromechanical systems and are equipped with the capability to monitor their state and generate fault messages in response to anomalous conditions of varying severity. Since removing a locomotive from a track for repair (or powering down a gas turbine or removing an airplane engine from wing) is an extremely expensive and disruptive procedure, it is desirable that

1. Problems occurring on the equipment while in operation are accurately identified so the repair can be scheduled best keeping with the severity of the problem.
2. Problems with the equipment are completely identified so the time in the repair shop is utilized at not merely fixing one problem but releasing an overall healthy machine.

ICARUS was designed to reason with the fault codes generated by locomotives during operation. In addition, it was a requirement ICARUS be able to build up it's information base quickly for rapid deployment and have the capability to learn as new information became available. It was also required that the tool be applicable across locomotive models and fleets with little modification.

This project presented three challenges that are typical of the real-world requirements deviating from textbook theory. First, diagnostic cases for the case base were not readily available and had to be reconstructed by mining historical repair records. Their accuracy was thus not guaranteed and case validation became an essential activity in itself. Very limited expert knowledge and time was available to fully validate the cases. Association of fault codes to specific repairs was difficult due to the standard railroad practice of multiple repairs on the same visit as well as some uncertainty about the accuracy of the date of repair. Finally, the continual nature of the fault logs made casting the case as a finite feature vector almost impossible.

This paper first presents the current operating scenario and the nature and availability of the data. We then present details of the process for generating meaningful cases. A new feature extraction and weighing algorithm is described as well as the results obtained from its implementation. This work is then related to prior activity in the literature. Finally some lessons learned from the project about CBR design and deployment are discussed.

2 Overview of Current Process

Locomotive fault logs are accumulated on-board the locomotive and are periodically uploaded to a database for access in case a diagnostic need arises. Highly skilled field engineers at General Electric Transportation Systems have acquired expert knowledge over time that enables accurate diagnosis of locomotive problems from an examination of the fault log. While this provides positive evidence for the diagnostic significance of fault logs, the volume of logged data makes it impossible to rely on human examination alone for reliable and consistent identification of locomotive problems on many hundred locomotives on a daily basis.

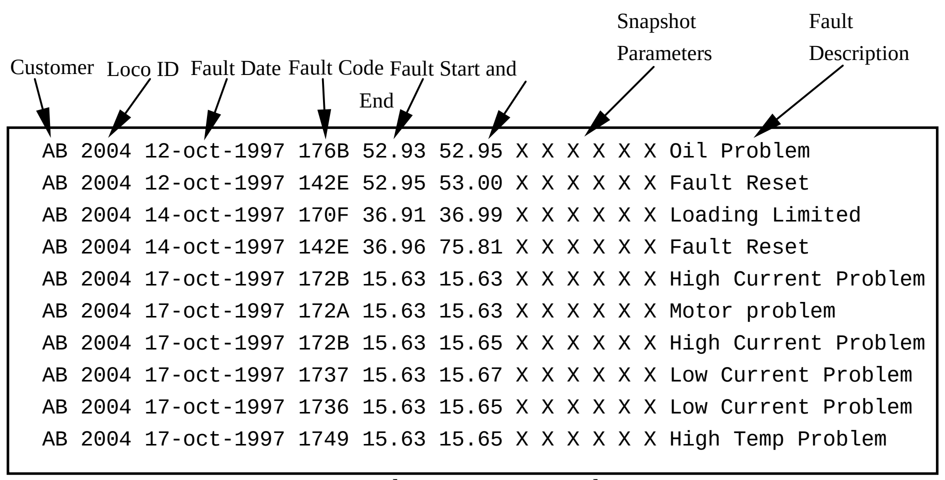
A case-based approach was considered as this appeared to be cognitively closest to the procedure used by the experts during diagnosis. It was desirable to move away from a rule based approach for several reasons. Some of these may be outlined as :

1. Accurate rules only existed for a small percentage of the locomotive's failure modes. The rest apparently happened in a manner too varied to capture in rules.
2. Frequent configuration changes and upgrades make rule based approaches hard to maintain.
3. Capturing knowledge as cases appeared to be the best approach for maintaining knowledge in a remote diagnostics environment where the diagnostic personnel were not necessarily all experts.
4. There was a realization that many more patterns may exist in the fault log data then anyone was aware of or could create rules for. There was a push for a learning approach for identifying these from case data.
5. It was desirable to deploy a functional system fairly rapidly, from concept to pilot operation in less than a year. However, experts had severe time constraints while there was considerable historical data that could be potentially mined for cases.

The objective of the project, thus was to build a tool that could take the fault log shown in Fig. 1 as input and output the top n repair codes with associated confidence values.

2.1 Historical Fault log and Repair Data

A sample fault log is shown in Fig. 1 . While the actual data constituting the log has been changed or masked, the essential features of the log are present. The first two columns identify the company that is operating the locomotive and the specific locomotive ID respectively. The next column indicates the date on which the fault occurred. The fault itself is identified by a 'fault code' a unique alphanumeric label associated with that fault. Next , a time stamp indicates the beginning and end of the fault message. The next few columns marked with 'X' represent a 'snapshot' of some important operating parameters of the locomotive at the time the fault occurred. Representative examples of such parameters would be speed, operating temperatures, pressure readings, whether certain switches are on or off etc. Finally a short text description associated with the fault code is displayed.

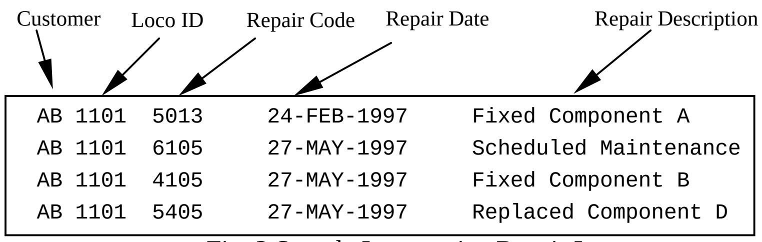

Customer
Loco ID
Fault Date
Fault Code
Fault Start and End
Snapshot Parameters
Fault Description

```
AB 2004 12-oct-1997 176B 52.93 52.95 X X X X X X Oil Problem
AB 2004 12-oct-1997 142E 52.95 53.00 X X X X X X Fault Reset
AB 2004 14-oct-1997 170F 36.91 36.99 X X X X X X Loading Limited
AB 2004 14-oct-1997 142E 36.96 75.81 X X X X X X Fault Reset
AB 2004 17-oct-1997 172B 15.63 15.63 X X X X X X High Current Problem
AB 2004 17-oct-1997 172A 15.63 15.63 X X X X X X Motor problem
AB 2004 17-oct-1997 172B 15.63 15.65 X X X X X X High Current Problem
AB 2004 17-oct-1997 1737 15.63 15.67 X X X X X X Low Current Problem
AB 2004 17-oct-1997 1736 15.63 15.65 X X X X X X Low Current Problem
AB 2004 17-oct-1997 1749 15.63 15.65 X X X X X X High Temp Problem
```

Fig 1. Sample Locomotive Fault Log

The fault log represents an arbitrarily long data stream with new data flowing in every day. Locomotive experts intimately familiar with the engineering specifics of the locomotive are able to look at the log alone and usually identify patterns that may indicate problems. It is worthwhile noting that multiple problems may be occurring simultaneously in the locomotive that will register a presence in the fault log. However, not all problems are critical enough to stop the basic operation of the locomotive and these accumulate till they are attended to during a regular service visit.

Since there was no direct association from individual fault codes to actual locomotive problems requiring repair, this information was sought to be implicitly acquired. A source of data used for this purpose was the repair database maintained by the manufacturer. The nature of the repair log was as shown in Fig. 2.


Customer
Loco ID
Repair Code
Repair Date
Repair Description

```
AB 1101  5013     24-FEB-1997     Fixed Component A
AB 1101  6105     27-MAY-1997     Scheduled Maintenance
AB 1101  4105     27-MAY-1997     Fixed Component B
AB 1101  5405     27-MAY-1997     Replaced Component D
```

Fig. 2.Sample Locomotive Repair Log

. The basic operation of the tool required taking the fault log as input and recommending a repair action with associated confidence. For reference, there were about 600 distinct faults that could be logged and 700 repair actions that could be taken. In addition, the repair actions could be logged in the database up to a week later than the actual physical repair.

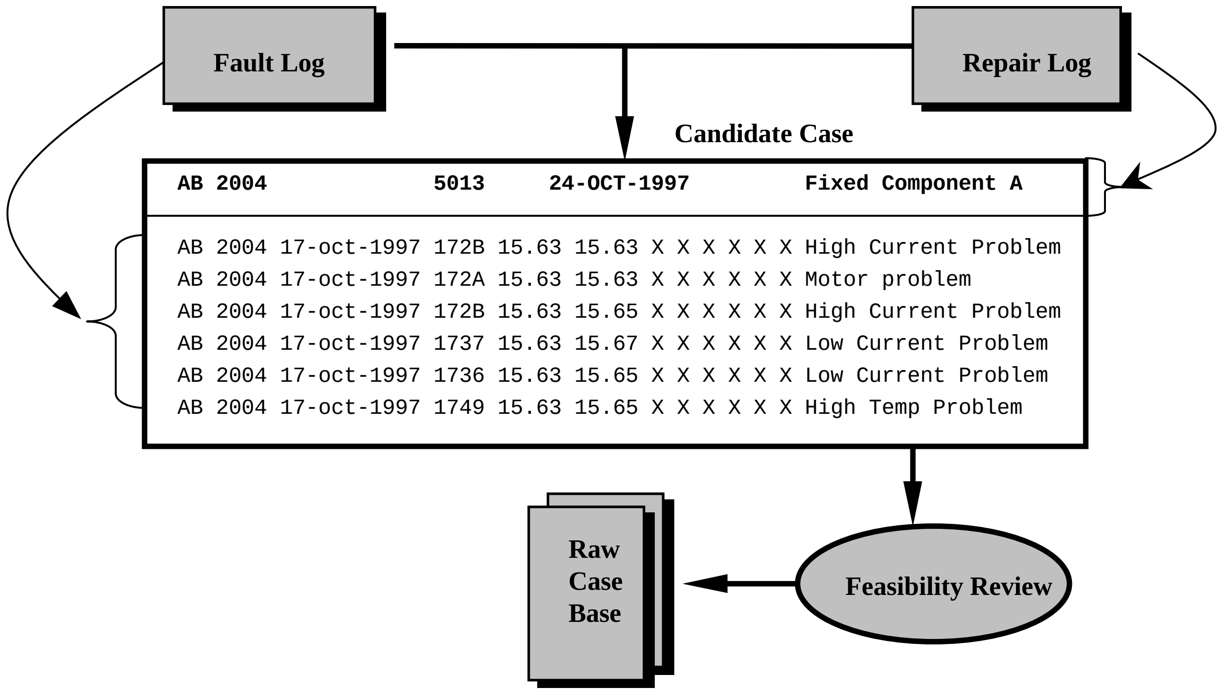
An approach was defined wherein, candidate cases would be generated to ‘seed’ the case base by mining historical fault log and repair records. These cases would then be minimally validated by format checking and checking for missing data. It was acknowledged that many of these cases could be diagnostically partially or completely incorrect. This may be due to the fact that either an incorrect repair was performed that did not actually address the problem that was causing activity in the fault logs or that the repair was incorrectly dated, or that there were multiple problems not all of which were addressed.

3 Process for defining and acquiring cases

The first task was to build candidate cases from historical fault log and repair records. A program was written to interleave the repair log with the fault log. A two year data window was chosen for prototype case generation with data gathered for over 200 locomotives. The process of raw case generation was as follows :

1. A particular repair type (diagnosis) was chosen for case collection.
2. All locomotives repair records were sequentially scanned for occurrence of that repair.
3. Every time that repair was encountered on a locomotive, a case was generated that contained the fault log contents for the N days preceding that repair.

This process is shown graphically in Fig. 3.



```
AB 2004            5013     24-OCT-1997         Fixed Component A

AB 2004 17-oct-1997 172B 15.63 15.63 X X X X X X High Current Problem
AB 2004 17-oct-1997 172A 15.63 15.63 X X X X X X Motor problem
AB 2004 17-oct-1997 172B 15.63 15.65 X X X X X X High Current Problem
AB 2004 17-oct-1997 1737 15.63 15.67 X X X X X X Low Current Problem
AB 2004 17-oct-1997 1736 15.63 15.65 X X X X X X Low Current Problem
AB 2004 17-oct-1997 1749 15.63 15.65 X X X X X X High Temp Problem
```

Fig. 3. Process for Case Acquisition

Each case was labeled by the repair code – the intended diagnosis. Multiple cases were collected for each repair code to capture the different fault code scenarios leading to that repair being the diagnosis. This process was repeated for many repair codes for which case-base coverage was required.

Each case so produced was still beset with certain problems. These included

1. Insufficient or missing fault log data : Since the fault log is fairly continual in nature, gaps of many days in the log indicated missing data rather than absence of faults.
2. Multiple repairs on the same day : There were many instances of 3-5 repairs performed on the same day. By our process, this resulted in many cases with identical fault logs but associated with different repairs.
3. Overlapping fault logs for repairs : If there were multiple repairs within N days of each other, they shared a common portion of the fault log for those N days.

For the above 3 conditions, heuristics were used to weed out cases that could possible 'contaminate' the case base. Cases with missing fault log were eliminated. Due to large quantities of historical data available, it was possible to restrict case selection to only those instances where there was only one repair on a given day. Nothing was done to correct for situation 3 since it was reasoned that the effect of overlapping fault log would be reduced with appropriate case feature weighing if sufficient cases for each of the overlapping repairs were collected.

Each case thus collected was subjected to a feasibility review by an experienced field engineer. The task of the field engineer was to review and eliminate any case in which the fault log was obviously **not** related to the repair performed. This task of saying yes or no placed a lower cognitive burden on the experts as compared to verifying each case as a 'gold' standard for that kind of repair. About 500 cases were collected following this procedure.

4 Feature Extraction

Much CBR work has implicitly assumed the availability of a finite number of indices by which to characterize a case. This is not always true, however as evidenced by the ultrasonic rail inspection system application reported in (Jarmulak & Kerckhoffs, 1997). This was certainly not true of our basic case structure. The N day fault log constituting each case could contain from zero to an indeterminate number of occurrences of each fault code. The total number of fault codes occurring in the case could potentially vary from one to over 700. It was evident that some feature extraction was necessary to identify indexes that would unify the case representation and make case matching possible. There were a number of options for feature representation. Cases could be matched or distinguished by taking into account

1. Presence/Absence of fault codes.
2. Fault code frequency
3. Combinations of fault codes
4. Time based trends in fault code occurrences i.e. if fault code frequency increased leading up to the repair.

5. Anomalous indicators in the parameter data i.e. if any of the continuous parameters were out of specification.
6. Sequence information in fault code occurrence i.e. if fault codes repeatedly occurred in a certain sequence .

However, there was another consideration constraining the choice of features. This was the fact that our cases were constructed by assuming that a certain causal relationship existed between fault logs and repairs data when their time line was overlaid. There was no initial evidence as to the degree of error associated with the assumptions underlying this approach. For this reason, it was decided to keep feature extraction fairly basic till such a determination could be made.

4.1 Fault Cluster Generation

Fault combinations were selected as the feature of choice for case representation. A variety of tests were carried out on historical fault log data to determine the maximum number of fault codes that appeared to occur repeatedly in combination on a given day. The analysis appeared to indicate that more than four faults seldom occurred repeatedly in combination in test data. As a result, the following approach was adopted. Each case was polled for a list of distinct fault codes occurring before the repair with which it was associated. The list of distinct faults was used to generate combinations (or fault clusters, as we termed them) as follows :

Distinct faults contained in case 1 : A,B,C,D
1-Clusters : A, B,C,D
2-Clusters : AB, AC,AD,BC,BD,CD
3-Clusters : ABC,ABD,ACD,BCD
4-Clusters : ABCD

This process was carried out for all the cases. A master list of fault clusters of each size was maintained. Each case was now indexed in terms of its features – namely Diagnosis + fault clusters. An example is shown in Fig. 4.

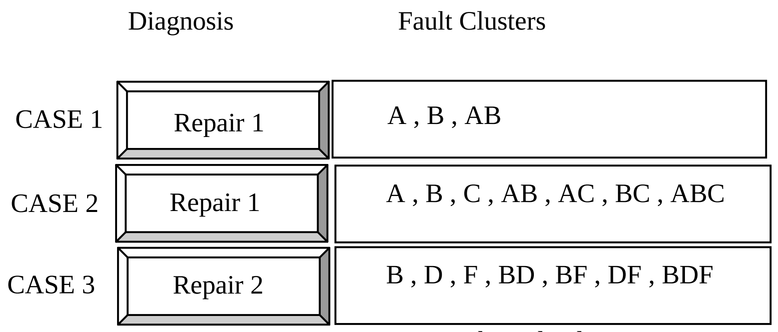


Fig 4 Case Representation with Fault Clusters

The objective of this exercise was to generate a complete list of candidate fault clusters of size four or less. Using this list, the next step was to determine which of this exhaustive list of clusters represented valuable repetitive patterns with diagnostic significance. This computer-intensive approach was adopted since there was virtually no expert opinion available to guide the selection of diagnostically useful fault patterns. Considering all possibilities of size four and under let the weighing algorithm consider a wide variety of cluster candidates in a reasonable time period.

4.2 Fault Cluster Weighing

Due to the inexact case creation procedure as well as the knowledge that there was not a one-to-one mapping between faults in the fault log and repairs, a process was created to assign weights to fault clusters based upon the cases in the case base. As new cases were continually being added, the system was designed to operate in two modes. In the learn mode, it calibrated the significance of available fault clusters based upon all the cases in the case base. During the diagnose mode, it used the weighted clusters as indicators to match the features of the incoming case with the most appropriate stored case.
The learn mode involves learning a weight value $\in [0,1]$ for each fault cluster. The weight is intended to be representative of each cluster's ability to isolate a specific repair code. If a cluster only appears in cases of a specific repair code, it has a weight of 1. On the other hand, if that cluster occurs with an evenly distributed frequency in cases of multiple repair codes, it's weight is appropriately lowered. A cluster is required to repeat a certain number of times before it is assigned a non-zero weight. After weight assignment, a clusters below a certain weight threshold are assigned a weight of zero. The process is shown in pseudo code below.

```
Program Calculate_cluster_weights

for each target repair code I (where repair code is a
categorization of a repair action)

 {
select distinct fault codes where (incident date -
fault date) < N days.

Store as case.

Delete from cases where #fault codes < min or > max.
}
for each case $C_I$
{  from  distinct fault list $F_i$ belonging to $C_I$
for (j = Maxclustersize ; j=1; j=j-1)

 [ Maxclustersize = 4 in our application]

     create distinct Fault clusters of size j .
}

For each fault cluster $F\_cluster_I$
{
count total # of cases it occurs in
count # incident codes it occurs in with what frequency.
Cluster significance = $f$(#cases,#distinct repair codes
it occurs in cases of,discriminating power)

If total # of cases < case_threshold : delete cluster
```

Generating [1-4] sized combinations of all faults in a case results in generating a lot of candidate fault clusters. In practice, thresholding resulted in only a small percentage of fault clusters emerging as significant in that they had non-zero weights. This was consistent with our approach of examining a wide variety of options to learn the features that were significant. In general, the number of single and double fault clusters that emerged as significant was larger that the number for three and four sized clusters. The average weight though followed the inverse relation. Three and Four sized clusters had higher average weights that one and two sized clusters. When faults appeared repeatedly in three and four sized combination they were usually strong indicators of diagnostic significance. The weight assignment can simply be recognized as the maximum conditional probability of any repair for a given fault cluster over all the repairs it occurrs before.

$$\text{Weight of Fault Cluster F\_cluster}_i = \text{Max}_j \left[\, P(\text{Repair}_j \,/\, \text{F\_cluster}_i) \,\right]$$

5 Case Matching

Once weights for fault clusters were acquired, case matching was straightforward. New diagnosis was requested by identifying the locomotive that was experiencing problems. The fault log database was queried for fault codes occurring in the N days preceding the diagnosis request.

Degree of match between a new and stored case was calculated as

$$\frac{[\sum \text{Weights of common clusters between stored and new case}]^2}{[\sum \text{Weights of Clusters in stored case}] \times [\sum \text{Weights of Clusters in new case}]}$$

The repair code associated with the case with the highest degree of match was the diagnosis returned by the system.

6 Case Validation

The case base currently contains cases for diagnosing over 50 repair codes. The number of cases associated with each repair code varies from three to seventy. Leave-one-out testing was performed to test the performance of the case base. In this process, one case was removed from the case base and formed the testing set. All other cases, as part of the training set were used to learn fault cluster weights. The case-base was then used to match and retrieve the top three repair codes in response to the left-out case. If the repair code associated with the testing case was in the top-3 set, the diagnosis was declared a success. This process was repeated with every case in the case base being the testing set once.

The accuracy of the case base was then tabulated as success % by repair code. There were a few repair codes where the case-base was consistently unable to correctly diagnose even in top 3 predictions more than 10-15% of the time. The cases associated with these repair codes were referred back to the domain experts. In most cases it was discovered that the repair codes were such that the fault codes could not be expected to predict them. In other instances, the same repair situation was classified under three different repair codes. Once these were unified, the accuracy of diagnosis on that repair code increased.

This process of case validation was a necessary closure to our initial approach of gathering cases that were approximately accurate. This allowed us to avoid requiring the time of domain experts to verify each case in the beginning. Now only a focused number of cases were required to be examined that did not appear to be consistent with each other.

7 Results of Experiments

Accuracy was measured only for repair codes that had over 10 cases associated with them. After removing repair codes deemed undiagnosable through the process in the previous section, accuracy on repair codes ranged from 23% to 94%. Overall accuracy was around 80% , assuming that the correct diagnosis appearing in the top three diagnoses given by the system was regarded as a success. In general the following trends were observed.

1. Repair codes associated with a greater number of cases had a higher diagnosis accuracy rate.
2. Accuracy increased as fault clusters of increasing size were used. The relative increase in accuracy was small but significant. Accuracy using fault clusters of size 1 was about 60 - 65%.

8 Related Work

There are quite a few examples of CBR being applied to diagnosis. We discuss a few that have addressed problems similar to ours both from an application as well as design viewpoint. Jarmulak et al. (1997) present a system that uses CBR for a rail inspection application. Their system uses image data as input and shares the limitation in that it is not easily expressed as a feature vector. They use a hybrid rule + case-based system for image classification with no adaptation. They also mention the need to periodically identify cases in the case base that never match well as possibly ‘bad’ cases or noisy images. Acorn and Walden (1992) report on SMART – a CBR help-desk system developed for COMPAQ. In contrast to our semi-automated, mining approach to creating cases, this describes a more conventional case-population process wherein senior engineers were designated as casebuilders with a daily review process. Correctness of the cases is not an issue. However, their observations that they could go live before having a complete and correct knowledge base is in agreement with our experience as well. Hennessy and Hinkle (1992) report on CLAVIER – one of the early commercial CBR applications. A key aspect motivating its development is described as the inability of the operators to articulate good rules. This, again, is consistent with our experience. CLAVIER's role as corporate memory that increases in quality and quantity with use very much in line with the role expected of ICARUS. Kitano et al. (1993) with their SQUAD system highlight the role of CBR systems as maintainers of corporate knowledge. Since they report dealing with over 20,000 cases, they describe a well managed human intensive process for case collection, filtering and quality control. They use a system of abstraction hierarchies to create neighborhood relationships between attributes. Interestingly, they also appear to use a system of combination generation. The motivation however seems to be to enumerate a sql type query for each type of attribute value below in the hierarchy from where the user has specified the attributes. Bonzano et al. describe an approach towards 'introspective' learning of feature weights in CBR, recognizing that standard CBR matching functions can be extremely sensitive to noise and irrelevant features and suitable weight vectors are not always available. While this concept is recognized in our approach, we do not make an effort to adjust feature weights in response to incorrect retrieval. This arises from our understanding that the error could lie with the case itself, and consequently consistent incorrect retrieval is used as an indication of a defective case rather than incorrect feature weighing.

CBR applications in similar domains have been reported under the INRECA project (Klaus-Dieter et al., 1995). INRECA uses induction to extract a decision tree to guide the user but uses CBR to handle unknown values. An application of INRECA to robot diagnosis uses a combination of causal rules, decision trees and weight factors for knowledge representation. This seems consistent with future development plans for ICARUS where a hybrid rule/case based system is eventually envisaged. In another application to CFM56 engines, use of legacy data to create initial cases with ongoing integration of model based knowledge is described. A concurrent benefit of the case building activity mentioned here is creation of a knowledge management process that helps highlight high occurrence failure modes through a systematic cleaning and analysis of data. This has been the case with ICARUS as cost-benefit analysis of fault generation from a diagnostic point of view is being revisited with a goal of generating more meaningful faults on the locomotive.

9 Institutional deployment and Cost-Benefit Analysis

ICARUS development was started in early 1997. A first prototype was deployed on pilot fault log data from 35 locomotives in October of the same year. A team of five diagnostics experts independently examines the fault logs daily and arrive at a diagnosis. These conclusions are compared with the output of ICARUS. This constituted the validation phase of the tool. An integrated recommendation was delivered to the railroad based on this activity. The primary benefit of identifying problems is that the locomotive can be better scheduled for repair and unscheduled failures leading to a mission loss are avoided. In most instances, a recommendation is kept open until feedback is obtained from the repair shop as to the actual work done. If this feedback corresponds to the top 3 repair recommendations of the CBR tool, the case is closed and declared as a success. If not, then it is classified as a failure. One field engineer at GE Transportation systems has been assigned the primary responsibility of running and validating the tool each day.

Both successful and failed attempts at diagnosis are examined in greater detail by the tool design team including the author. In many cases, experts point out that certain problems are not well manifested in fault logs and cases relating to these repairs are removed from the case-base. The primary driver for seeking expert input is when the case base is unable to predict a repair code at a > 50% accuracy despite having > 10 cases for it. This focused approach helps minimize expert time requirements.

Some early successes in diagnosing problems in the pilot program have helped management allocate increasing resources to the project. The biggest benefit is that new incoming cases (that are not mined from historical data but based upon daily analysis) are of much higher quality due to expert validation and have contributed to increasing the accuracy of the tool. It is estimated that a savings of a few thousand dollars could be realized per locomotive per year just by optimizing its repair actions based upon an accurate understanding of the failures occurring on board. Over 350 locomotives expected to be monitored in 1999-2000, this adds up to a considerable sum. There is a considerable productivity benefit as well as a limited staff of upto ten experts will be required to monitor the 350 locomotive fleet and tools like ICARUS can considerably reduce the effort required for diagnosis.

10 Discussion

A number of lessons were learned in the course of this project. Some of these may be listed as

1. The availability of high quality cases cannot be taken for granted. In complex domains specially, it becomes increasingly difficult to find expertise that will certify cases as fully correct. In this application we were specially mandated not to rely on ‘preconceived’ expert knowledge to guide the case base development – rather to learn from the data. Many experts freely acknowledged that there was possibly much more hidden in the data than they had expertise over.

2. Case representation can be a design issue. In most case-based applications, the identification of a case feature vector arises naturally from the way the case exists. Out of many possible features that could characterize the cases in this application, fault combinations were one of the features identified as being potentially significant and were chosen for case representation. Feature

weighing was able to consider and eliminate a majority of fault clusters as being diagnostically insignificant.

3. The assumption that each case contained data associated with only one diagnosis was not valid in this case. The extent that multiple simultaneous problems were being manifested in the fault codes was not known. Again, this was addressed partly by choosing candidate cases carefully and subsequently by feature weighing. Only fault clusters that consistently occurred in cases of a particular repair code were assigned high weights by the feature weighing algorithm.

4. Commonly recognized advantages of a case-based reasoning approach like quick deployment, capability for continuous learning, low knowledge elicitation needs and a measure of explanation stood true in this application. The system was developed in about eight months and is currently running on live pilot data. While the initial case seeding was based on historically mined cases, future cases that are being added are of much higher quality since the final resolution of problems will are accurately tracked and verified from the field.

   The portability of the case-based approach was vividly demonstrated once the system was developed. Business focus required that ICARUS be applied to a model of locomotives different from the data on which the system was developed. In less than a month, new seeding cases were acquired, feature weights were recomputed and another version of ICARUS was released. This was in comparison to a rule-based approach which would have required development from scratch. For reference, previous rule-based approaches for locomotive models had taken few years to develop.

   Incremental learning was specially important in this application. Locomotives undergo frequent hardware and software changes. A case-based approach could adapt to this, given sufficient quantity of cases.

5. Learning weights for case features can help make implicit knowledge explicit. In many cases, a physical explanation could be attached to particular faults occurring together. This provided an additional means to occasionally check the knowledge in the case base.

6. As yet, there is no concept of adaptation in the working of ICARUS.

7. Finally, the ability to come up with a working albeit incomplete system early on was vital in maintaining management and user involvement in the application.

11 Conclusions and Future Work

ICARUS will be deployed for providing monitoring support to over 350 locomotives in 1999. As more cases are added to the system, we are exploring additional features by which to characterize cases to sustain and improve the system's accuracy. Preliminary results incorporating fault occurrence trends as case features have shown evidence of positively impacting diagnostic accuracy. As can be expected, this too has a stronger effect on certain repair codes as compared to others.

In conclusion, we have presented a case-based application for diagnosis that employs a variety of pre and post-processing techniques to transform historical data into a format suitable for a case-based approach. We present an 'propose and verify' approach towards case generation where candidate cases of approximate diagnostic accuracy are generated and case performance metrics are used to isolate cases that may need expert validation. Feature weights are learned from the data. The application is such that a complete and accurate diagnostic formulation with any approach is practically impossible. Our experience has shown that a case-based approach has been able to contribute significantly towards capturing a tractable amount of knowledge even if approximately, and consequently reducing the load of the diagnostics expert.

Acknowledgements

The author would like to acknowledge and thank Nicholas Roddy for developing the CBR software as well as running validation tests. Thanks are due to Tom Shaginaw for motivating the Case-Based approach. Special thanks to David Gibson for supporting our efforts with business and data support and domain expertise.

References

1. Jarmulak, J., Kerckhoffs, E.,Veen, P. : Case-Based Reasoning in an Ultrasonic Rail-Inspection System. In: Leake, D., Plaza, E.(eds.): Case-Based Reasoning Research and Development. Lecture Notes in Computer Science, Vol 1266. Springer-Verlag New York (1997) ,43–52.

2. Acorn,T., and Walden, S. : SMART: Support Management Automated Reasoning Technology for Compaq Customer Service. In Proceedings of AAAI-92. Cambridge, MA: AAAI Press, MIT Press.(1992)

3. Hennessy, D., and Hinkle, D. : Applying Case-Based Reasoning to Autoclave Loading. IEEE Expert,(1992), 7(5), 21-26.

4. Kitano, H., Shimazu, H. and Shibata, A. : Case-Method: A Methodology for Building Large-Scale Case-Based Systems. In Proceedings of the Eleventh National Conference on Artificial Intelligence, (1993), 303-308.

5. Bonzano, A., Cunningham, P. and Smyth, B. : Using Introspective Learnng to Improve Retrieval in CBR : A Case Study in Air Traffic Control. In: Leake, D., Plaza, E.(eds.): Case-Based Reasoning Research and Development. Lecture Notes in Computer Science, Vol 1266. New York (1997) ,291–302.

6. Klaus-Dieter, A., Auriol, E., Bergmann, R., Breen, S., Dittrich, S., Johnston,R. Manago, M., Traphoener, R., Wess, Stefan : Case-Based Reasoning for Decision Support and Diagnostic Problem Solving: The INRECA Aproach. In B. Bartsch-Sporl, D. Janetzko & S. Wess (eds.), Proceedings of the 3rd workshop of the German special interest group on CBR , (1995), 63-72.