Layers of Memory and Reasoning in the Problem of Diagnosis in Manufacturing

Radhika Selvamani B., Deepak Khemani
Indian Institute of Technology Madras
Computer Science and Engineering Department
bradhika@cse.iitm.ernet.in, khemani@iitm.ac.in

Abstract

Any system whether it is a machine, a human being or an organization, has several layers of functionality. In an organization these layers need to communicate to exchange information thus facilitating co-ordination and co-operation to perform a particular task in a domain. In this paper we look at the problem of diagnosis in a manufacturing setting to observe the several layers of processing in the domain and to semi-automate the same in a computer. We start with describing the problem of diagnosis in the manufacturing setting and the various goals to be achieved by various personnel in the organization. We propose a layered knowledge architecture in the domain to carry out the goals of these individuals by capturing and creating knowledge in a computer and operationalizing the acquired knowledge in the domain.We focus on one of the subgoals of adapting defective instances to provide defect free cases dealt by the Case Adaptation algorithm. A CBR system simulates a domain expert to provide feedback on the utility of the algorithm. Experiments support that the architecture can improve product quality by completing from memory the knowledge cycle in the manufacturing setting.

1 Introduction

Organisational systems have internal structures that mediate roles and relationships among people working toward some identifiable objective [Bennet and Bennet, 2003]. The task of knowledge management is to capture explicit and tacit knowledge of an organization to facilitate the access, sharing and reuse of that information [Dieng-kuntz and Matta, 2001]. Case Based Reasoning is a memory based reasoning methodology which functions by the principle that similar problems have similar solutions [Kolodner, 1993]. CBR has been used widely as a knowledge management tool in various applications [Watson, 2003][Khemani *et al.*, 2002]. Hybrid CBR systems evolve with a goal to efficiently capture specific knowledge in the form of cases, to elicit or create general knowledge in the form of rules and to apply them in the domain. A framework to reuse the captured experince in the domain and to refine the observations would close the Case-Based Reasoning cycle in the domain [Aamodt and Plaza, 1994]. An application of the Case Based Reasoner in closing the manufacturing loop has been discussed by Peglar [Price C. and A., 1997]. There are several layers of processing involved in knowledge management in a computer as in Fig.1. These layers need to communicate with each other in order to accomplish a task assigned by a user to the computer. If the communication among the layers get interrupted it is not possible to perform the required task using the computer. An intelligent user is one who is aware of these layers, their mode of communication and their computational limitations. Hence it is possible for an experienced user to accomplish the required tasks more efficiently.

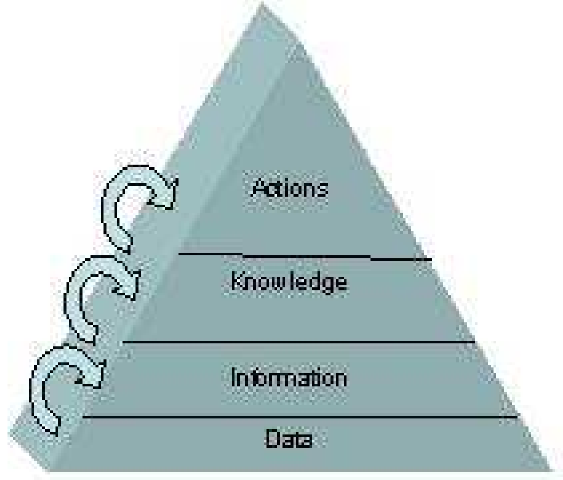

Figure 1: Knowledge Pyramid [Aamodt and Nygard 1995]

Another example is that of a human being learning to drive a car. Initially a novice learner has to consciously control his responses at several layers of perception of the road and traffic to drive the car. As he gains experience very few voluntary efforts are required to controle the car. Other knowledge requirements are taken care of involuntarily by some higher layer in the person, which he is not always conscious. We have chosen to address the problem of diagnosis in a manufacturing setting to observe the several layers of processing in the domain and to semi-automate the same in a computer. We begin by describing the problem of diagnosis in the manufacturing setting and the various knowledge layers observed in section 2. We propose a layered knowledge architecture with three objectives to close the knowledge cycle which prevails among an analyst, shopfloor personnel and an experienced expert in the domain. We use a case-based reasoner to simulate a domain expert who would provide feedback to the various knowledge activities in the domain. We focus on one of the subgoals, case adaptation dealt by the CA algorithm in section 3. Experiments on the CA algorithm have been discussed in Section 4. Results and conclusions are discussed in section 5.

2 Diagnosis in Manufacturing

Diagnosis is the process of identifying the cause for the discrepancies observed between the actual outcome and the expected outcome in a system. In a manufacturing domain the problem of diagnosis is identifying those process parameters, which cause defects in a product.

2.1 Case-Based Reasoning and Adaptation

In case-based reasoning specific knowledge describing a context is called a case. The cases are stored in the case base. The three important knowledge containers in a CBR system include the case vocabulary, reuse-related knowledge and the case base as introduced by Richter[Richter, 1995]. Previous experience is captured in the form of cases, which are stored as problem solution pairs in the case base.The classical case-based problem solving cycle is as shown in Fig.2. A case that is similar to the current problem is retrieved from the case base. Then the solution contained in this retrieved case is reused to solve the new problem i.e. the solution is adapted in order to come to a solution of the current problem. The adapted case is revised by the user and then applied to solve the problem at hand. The revised cases along with the expert feedback are retained for reuse.There are two types of reuse-related knowledge in CBR leading to tranformational adaptation and generative adaptation of case knowledge. Transformational adaptation relies on a set of adaptation rules or operators thet describe how differences in the problem lead to required modification in the solution. On the other hand generative adaptation methods require a complete generative problem solver that is able to solve problems based on general knowledge, i.e., without using any cases at all. While dealing with the problem of diagnosis in manufacturing we store the process details as well as the defect description as cases in the case base. We use k-NN algorithm for retrieving a case matching a query. Cases in the case base store the strength of good and bad instances of their application in the domain. Whenever a case retrieved for a query has a high probability of causing defects, we use the decision tree built from the domain data to adapt the defective query instance generating a new case which needs to be tried in the domain. This new generated trial case is stored in the short term memory till it matures with repeated usage to become an experience in the longterm memory.

2.2 Knowledge Discovery

The process of knowledge discovery considers the entire process from the onset of data to the generation of knowledge

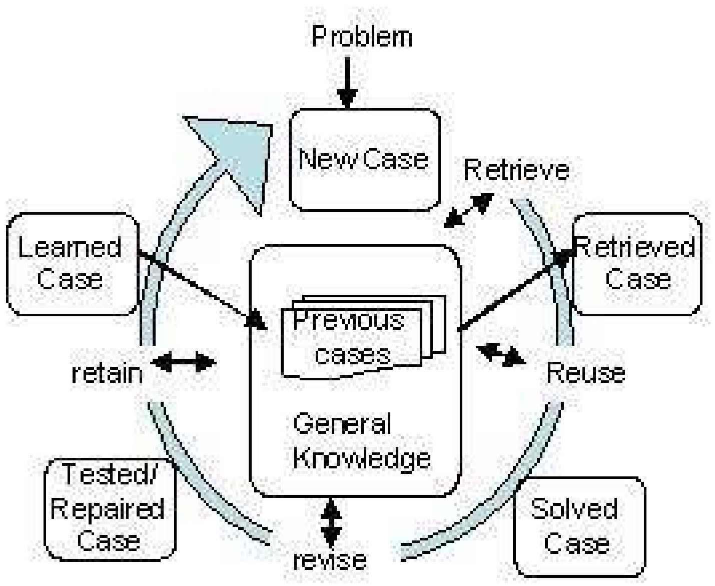

Figure 2: CBR Cycle [Aamodt and Nygard 1995]

[Chih-Ping Wei and J.Shaw, 2003]. It consistes of five phases, including selection, preprocessing, transformation, data mining and interpretation/evaluation. The selection phase aims at selecting a data set or focussing on a subset of variables or data samples from various data bases, on which the knowledge discovery is to be performed. After the selection phase, the data needs to be preprocessed. The preprocessing stage includes integration of data from multiple sources, cleaning the data for eliminating inconsistency and handling outliers. Then the missing values are handled either separately by different imputation methods or by the knowledge discovery algorithm. The transformation phase transforms the preprocessed data into a particular representational form as required by the data mining technique to be used. Finally the discovery algorithms extract the knowledge patterns that exist in the data. We use the decision tree induction technique, a supervised learning method that constructs a decision tree from a set of training instances to obtain the general knowledge required for adapting a defective instance. We use c4.5 [Quinlan, 1992] algorithm to build a decision tree. The c4.5 algorithm uses information theoretic approach to build a minimal tree to classify an instance. The attributes are evaluated using their information gain to select the best attribute for testing at each node.

2.3 Problem:Goals and Tasks in the Domain

The problem of diagnosis in manufacturing has several facets. Given the information regarding the domain attributes and the corresponding data records our goal is to improve the overall quality of the products manufactured in the domain. We need to obtain subgoals, which are more specific to the problems in the domain, the users who would be interacting with the system and the tools and techniques available in the domain. One of the major subgoals has been to identify the cause for each of the defects that occure in the product. The next subgoal is to adapt defective process instances to generate new diagnosed cases in order to improve the quality of the products, which may be stored in a short term memory. The third subgoal is storing the repeated cases with successful solutions and failures in a long-term memory, which acts as a case based reasoner. These three subgoals have been identified with the problem of diagnosis in a manufacturing domain. The first sub goal of identifying causes for defects have been solved using Quinlan's c4.5 algorithm wrapped in CIDTree (Cause Induction for Diagnosis Tree[Selvamani and Khemani, 2005]). The second subgoal of adapting defective process instances to create defect free cases has been discussed in this paper. Work is in progress to design the long term memory which would act as a Case Based Reasoner as well as decision maker in changing outdated processes that prevails in the domain documents.

3 Layered Knowledge Architecture for Diagnosis

In order to simulate human intelligence in a computer we need to define data, information, knowledge and wisdom as they are perceived by a human being. Any signal or symbol that is created in a domain is data. Data that has been interpreted by a person according to the context may be called information. The same data may be perceived differently by different individuals depending on the context, their capability and the task that the person intends to perform. Knowledge is nothing but the processed information which exists in a form which can be directly applied to the task at hand. Wisdom is the capability of a person to apply the acquired knowledge to solve a problem at hand. We observe the above four layers in the problem of diagnosis in a manufacturing domain.

3.1 Layers in Knowledge Creation

The different layers in the knowledge pyramid [Aamodt and Nygard, 1995] may be defined as follows in our context. **Data Layer:**The data fetched from the databases and the domain attributes, which form case vocabulary exist in the data layer. **Information Layer:** Creating domain vocabulary, associating domain vocabulary with the data attributes in the databases and assigning importance to the domain attributes are some of the actions that take place in the information layer. The description of the defect is also provided by the expert at the information layer. The information layer needs more interaction with the domain experts to elicitate the expert knowledge in to the machine. **Knowledge Layer:** Knowledge is information, which has been processed in such a way to accomplish a particular task. In case of classification, the decision tree is the knowledge source that exists with the computer. Similarly, the cases acquired in a case based reasoner together with the similarity function and the indexing mechanism are the knowledge sources that exist with the computer. Knowledge may be acquired directly from the domain experts, indirectly using acquisition tools or learnt from the data and information layer. **Wisdom:** Wisdom is applying the correct knowledge at the correct occasion to accomplish the user's task

The input data consists of integers, ordered data, unordered data and symbolic data[Balaraman and Vattam, 1998]. In order to improve the quality of the products in a refractory manufacturing domain, we start building the information layer based on the data that exists in the domain, guided by the knowledge that exists with the domain experts. We first obtain the domain vocabulary and their characteristics such as type, range, weight etc.,from the experts. We then associate the domain vocabulary with the data attributes to fetch the required information for further processing by the knowledge layer. We get the defect description from the experts. Fig.3 explains the layers of processing involved in diagnosing the defects in a manufacturing setup.

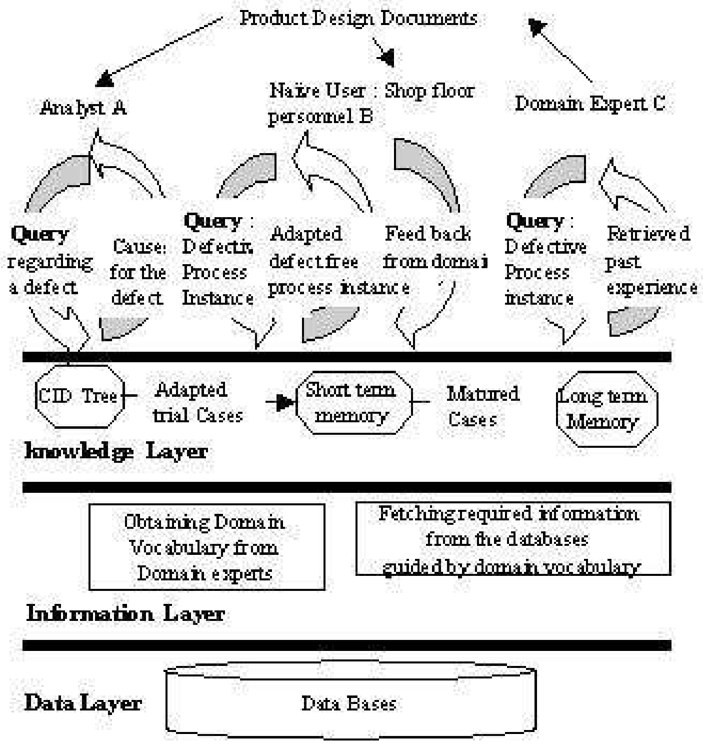
Product Design Documents
Analyst A
Naive User : Shop floor personnel B
Domain Expert C
Query regarding a defect
Cause for the defect
Query : Defective Process Instance
Adapted defect free process instance
Feed back from domai
Query : Defective Process instance
Retrieved past experience
CID Tree
Adapted trial Cases
Short term memory
Matured Cases
Long term Memory
knowledge Layer
Obtaining Domain Vocabulary from Domain experts
Fetching required information from the databases guided by domain vocabulary
Information Layer
Data Layer
Data Bases

Figure 3: Layers of Processing

3.2 Components in the Knowledge Layer

We perceive the components in the knowledge layer as the goals of 3 individuals in a domain (Fig. 3). Let A be an analyst in the domain, who has no previous experience about the defects in the domain. He has the data and the defect description. He aims to build a model of the domain from the existing data to accomplish diagnosis. Let S be a shop floor personnel who is capable of modifying the process based on observations from person A. Let E be a domain expert who has been working in the domain for years and has well abstracted, indexed cases stored in his memory as experience. His reasoning is similar to that of a case based system.

The first subgoal of identifying the cause for a specified defect is performed as follows. The decision tree created using c4.5 algorithm is wrapped in a CIDTree which has defect scores associated with each node in the tree. Each node in a decision tree refers to a test on an attribute. A path from the root to a decision node corresponds to a pattern oberved in the data. The defect score at a node conveys the relevance of that data pattern from the root to the node in causing the defect. This knowledge is created when the user wants to know the cause for a new defect which has not been solved by him earlier or when he wants to compare his experience with the machine's knowledge.

The second subgoal is adapting the process instances to create cases that would eradicate the defect. This step is needed to operationalize the knowledge that has been created in the previous step and has been approved by domain experts. Operationalization, in this contex means, allowing naive users in the shop floor to use the knowledge created in the knowledge layer. In other words, the knowledge created as CIDTree in the knowledge layer is transformed to the data layer in a form, which could be applied in the domain to obtain user feedback regarding the goodness of the case.

The third subgoal is the shopfloor employee communicating the consistent results back to the domain expert. The expert could store them in his longterm memory or documents in an appropriate format so that he could refer to them during the design phase of a product. This process could be supported by a case based reasoner with matured cases, that is cases that have been tried in the domain.

The short-term memory of S stores the cases created using A's models, applies them in the domain and stores the feedback. When a created case is consistently good or bad for a pre-specified number of accesses in the short-term memory the information is conveyed to E who decides to store the result in his long-term memory as well as in domain documents for an update in the design handbook. S is an individual who needs to communicate through all three layers, the data, information and knowledge to try a manufacturing process and provide feedback. Each of the individuals store knowledge in different forms in different types of memory providing different forms of inference and reasoning.

3.3 CIDTree

A decision tree used for classification has a single attribute tested at each node leading to the class of the instance being classified at the leafnode. CID tree is a decision tree with defect score for each node of the tree.The training data for the CID tree is prepared as follows. The process instances that have a particular defect are labeled “Defective” and the rest “NonDefective”. A decision tree is built for the data to discriminate among the above classes. A path from the root to a node in the tree represents a data pattern. A defect score is calculated for each node from the support and confidence of the corresponding pattern.

3.4 Case Adaptation

We are dealing with the second subgoal of adapting defective process instances in this paper. Assume that the knowledge model of the analyst has been implemented as the CIDTree in the computer. The analyst's aim is to adapt a defective instance to provide a defect free case. Analyst A should also check if it is worth building a model and adapting cases in improving the quality of the products. In the CIDTree the node with the largest defect score contains the knowledge to rectify the defect. Suppose node p has the highest defect score, then the sibling of node p is the closest matching non defective node. The Case Adaptation algorithm (CA Algorithm) works as follows (Fig. 4). The manufacturing details of a single product in the domain is considered a product instance. A process instance or a set of merged process instances stored in memory is called a case. A case instance to be processed through the expert system is called a query case. Let us assume that a query case has no missing values. The query case is parsed through the CID Tree, where a test at each node directs the path of the query towards the leaf. If a defective leaf node is reached the node with highest defect score on the path is marked as ”flawed”.The entire process is repeated from the sibling of the node with the highest defect score.This process is repeated till a non-defective leaf is reached. If we reach a non-defective leaf at the first pass then the query case has no changes else there is a change in the query case for every backtrack.

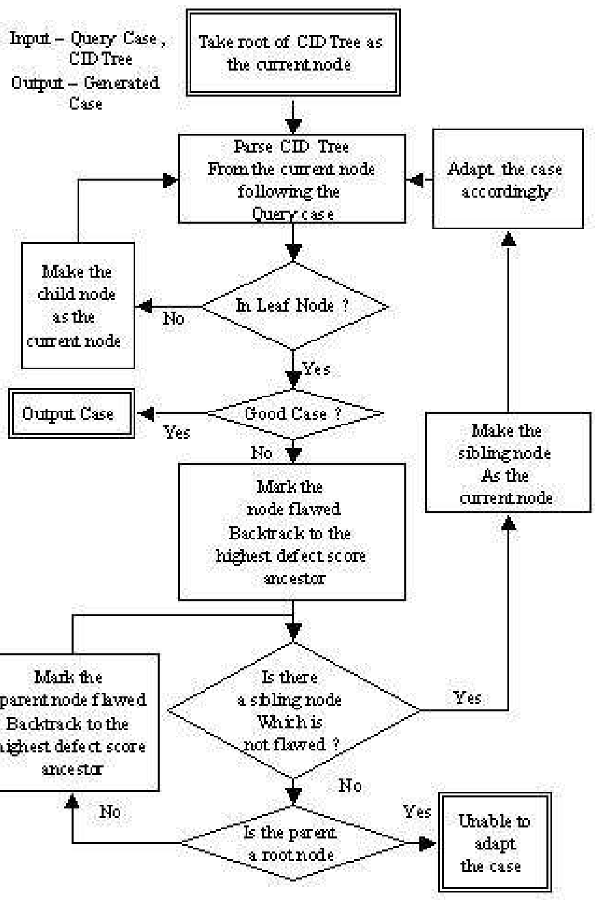
Input – Query Case , CID Tree
Output – Generated Case
Take root of CID Tree as the current node
Parse CID Tree From the current node following the Query case
Adapt the case accordingly
Make the child node as the current node
No
Is Leaf Node ?
Yes
Output Case
Yes
Good Case ?
No
Mark the node flawed Backtrack to the highest defect score ancestor
Make the sibling node As the current node
Mark the parent node flawed Backtrack to the highest defect score ancestor
Is there a sibling node Which is not flawed ?
Yes
No
No
Is the parent a root node
Yes
Unable to adapt the case

Figure 4: Case Adapatation Algorithm

4 Experiments

The utility of CIDTree has been manually evaluated by the domain experts and they found it convincing[Selvamani and Khemani, 2005]. The utility of CA algorithm has been evaluated by simulating the domain using a case based reasoner. The available process data in a particular domain has been partitioned into two parts, a set of query cases and a set of instances for the decision tree and the Case Base.

- Query set - defective process instances which are undiagnosed cases in the domain
- Simulator data - process instances used to build a simulator (Case based reasoner)
- Training data - process instances to build a CID Tree.

The partitions are made based on a predefined ratio (Fig. 5).

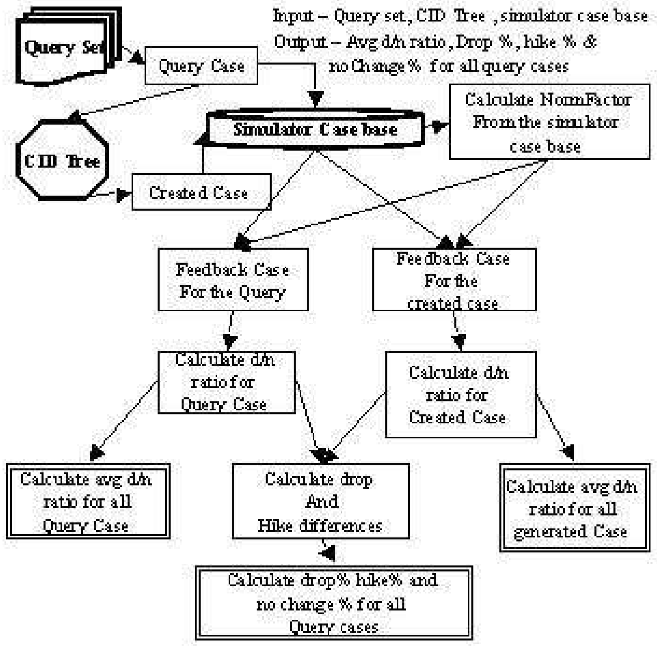
Input – Query set, CID Tree , simulator case base
Output – Avg d/n ratio, Drop %, hike % & noChange % for all query cases
Query Set
Query Case
CID Tree
Simulator Case base
Calculate NormFactor From the simulator case base
Created Case
Feedback Case For the Query
Feedback Case For the created case
Calculate d/n ratio for Query Case
Calculate d/n ratio for Created Case
Calculate avg d/n ratio for all Query Case
Calculate drop And Hike differences
Calculate avg d/n ratio for all generated Case
Calculate drop% hike% and no change % for all Query cases

Figure 5: Experimental Setup

5 Experiments

5.1 Building a Simulator

CBR has been chosen to simulate the domain, because theory and practice varies widely in the domain, hindering application of model based reasoners. Conducting real time experiments in the domain is quite costly. An instance is added as a new case to the case base only if it is dissimilar by greater than a pre-specified threshold to any other case in the case base. If an instance does not become a new case it is augmented to the closest matching case by updating its frequency of creating defective and good cases.Each case in the case base stores d_i the number of defective instances and n_i the number of non-defective instances

Table 1: Domain Data

Defects	Data	Defect	Query	Training	CB-Inst.	CBR
CrackFou	206	42	25	181	181	87
CrackFur	206	42	25	181	181	155
EC1Fou	206	56	17	189	189	99
EC1Fur	206	56	17	189	189	158
HT1Fou	206	75	41	165	165	83
HT1Fur	206	75	41	165	165	136
SpallFou	206	38	22	185	185	93
SpallFur	206	38	21	185	185	162
IntDefTube	253	50	28	225	225	225
JarringTube	248	33	14	234	234	234
ScratchStrips	1146	133	58	1088	1088	1027
DentPunchStrips	1146	48	30	1116	1116	1054
CracksStrips	1146	19	11	1135	1135	1073

5.2 Evaluating the Created Cases

Experiments have been carried out for diagnosing defects in three domains, the refractory blocks domain, the steel strips and steel tubes domains. Table 1 tabulates the details about the data used in the experiments. In the refractory blocks domain there are three shopfloors namely foundry(fou), furnace(fur) and finishing(fin). There are four different types of defects namely cracks, edge cracks(EC), hot tears(HT) and spalls. Each of these defects has been dealt separately for each of the shopfloors. The steel tubes-manufacturing domain provided data for two kinds of defects, the internal defect and jarring. Each of these data sets has about 200 process instances. The steel strips manifacturing has three defects to be diagnosed, the scratch, dentpunch and edge cracks each with about 1000 process instances. Refer Fig.4 for the CA Algorithm. The following steps are followed in measuring the peformance of the CA algorithm.

- Each undiagnosed cases in the query set is adapted to create a set of adapted cases.
- For each set of instances (the query set as well as the adapted set) a corresponding set of cases are retrieved from the simulator case base.
- The defect(d_i)/none(n_i) ratio is calculated for each instance in the query and the adapted set from the retrieved cases. The ratio is normalized
- The average defect(d_i)/none(n_i) is calculated for each set.

5.3 Parameters Measured to Evaluate CA Algorithm

Two important parameters defined to measure the utility of the CA algorithm are the avg DN ratio and the drop percentage. These are explained in the following section.

Normalized Defect/Non Ratio

This is used to measure how often a case has been successfully applied without the particular defect. Each case c_i stores the number of defective instances(d_i)and the number of non-defective instances(n_i) augmented to it.

The DN ratio of case c_i = d_i/n_i.

If the DN Ratio of a case is high the case is highly probable to cause defects. The DN Ratio needs to be normalized in order to compare different cases. Normalization is done by calculating the possible range of DN Ratio as NormFactor. NormFactor is calculated as follows.

Minimum possible d/n ratio :

$dn_{min} = 0$

Maximum possible d/n ratio :

$dn_{max} = infinite$ when $(n = 0)$.

Let us assume that the highly defective case has atleast one non-defective instance inorder to avoid inifinity in our calculation. Let the maximum numer of defects in a case in the simulator case base be sd_{max}

Maximum possible DN ratio :

$dn_{max} = sd_{max}/1$ when $n = 1$

Normalizing Factor :

$NormFact = dn_{max} - dn_{min} = sd_{max}$

Normalized DN ratio :

$dn_i = \frac{d_i * 100}{(n_i * NormFact)}$

Average Defect/Non Ratio

To calculate how probable is the current query set in causing a defect we need to find the average DN ratio of the cases retrieved for each of the query in the set. Let MaxQuery be the maximum number of instances in the query set. Average DN ratio of query set :

$dnq_{avg} = \frac{1}{MaxQuery} * \sum \frac{(d_i * 100)}{(n_i * NormFact)}$

Similarly the average DN ratio could be obtained for the set of adapted cases

Expected Result 1: We expect the query set to have higher average DN ratio than the adapted case set (Fig. 6).

Drop Ratio

Each query case in the query set has a corresponding adapted case in the adapted set. Drop ratio shows how many of the suggested cases have successfully reduced the DN ratio of the corresponding query case. Let MaxChanged be the number ofquery cases that have been altered during the process of adaptation. Let nDdrop be the number of query cases which have a adapted case with relatively less DN ratio. Let nDHike be the number cases which on adaptation had greater DN ration than the query cases. Let nNoChange be cases with no change in DN ratio on adaptation.

Drop percentage = $\frac{nDdrop * 100}{MaxChanged}$

Similarly we can calculate Hike percentage (how many of the suggested cases have raised the DN ratio and hence the probability of causing defects).

Expected Result 2: The Drop percentage should be higher than the Hike percentage (Fig. 7).We could fix an optimum threshold of 0.96 for the similarity of cases in the case base based on the above experiments.

6 Results and Conclusions

We have captured the domain information in the information layer and have successfully created the knowledge sources, the CID tree as well as a short term memory to store adapted cases in the knowledge architecture. It is observed that the adapted cases have lesser average defect(d_i)/good (n_i) ratio compared to the query cases. Hence we can conclude that cases adapted using CA algorithm on CIDTree are capable of improving the quality of the products in the domain. When we look at the effect of the algorithm on individual cases through the drop percentage and hike percentage we can infer that none of the adapted cases are worse than the query case i.e. hike percentage is near zero. Having measured the utility of the CA algorithm, our next goal is to create a long term memory that would store the diagnosed cases in such a way to be retrieved as and when a defect occures. The organization works by updating a design document when a reccuring defect is successfully rectified by a particular process change in the domain. The adapted cases which would reside in the long term memory would successfully lead to process rectification to close the knowledge cycle in the domain.

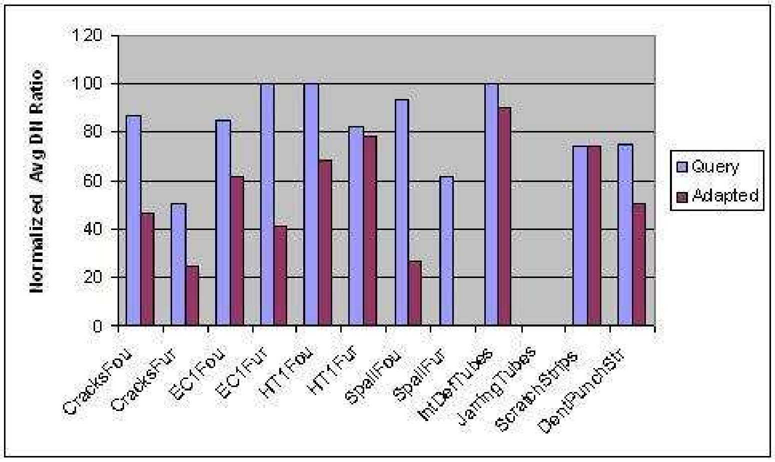

Figure 6: Data Comparing D/N ratio of Query Set and Generated Case Set

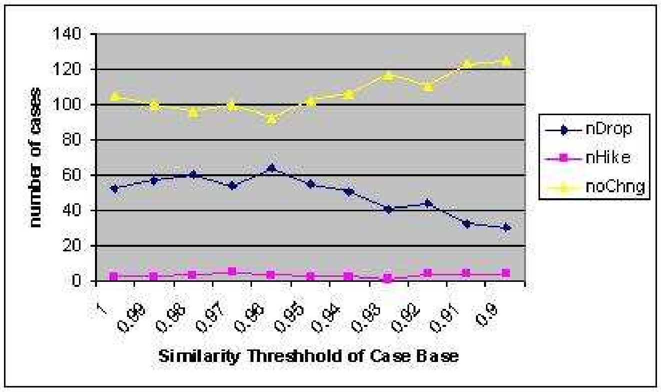

Figure 7: Data Comparing drop or hike in d/n ratio for varying threshold of similarity for the simulator casebase

Acknowledgements

We acknowledge the support of Murugappa Group of Companies Ltd., Chennai in funding the research projects as well as providing required data for the experiments.

References

[Aamodt and Nygard, 1995] Agnar Aamodt and M. Nygard. Different roles and mutual dependencies of data, information and knowledge. In *Data and Knowledge Engineering 16*, pages 191–222, 1995.

[Aamodt and Plaza, 1994] Agnar Aamodt and Enric Plaza. Case-based reasoning: foundational issues, methodological variations and systemic approaches. In *AI Communications 7(1)*, pages 39–59, 1994.

[Balaraman and Vattam, 1998] Vivek Balaraman and S. Vattam. Finding common grounds in case based systems. In *In Proceedings of the International Conference KBCS'98*, Indian, 1998.

[Bennet and Bennet, 2003] David Bennet and Alex Bennet. The rise of the knowledge organization. In *Handbook on Knowledge Management 1 Knowledge Matters*, NewDelhi, India, 2003. Springer (India) Private Ltd.

[Chih-Ping Wei and J.Shaw, 2003] Selwyn Piramuthu Chih-Ping Wei and Michael J.Shaw. Knowledge discovery and data mining. In *Hand Book on Knowledge Management 2 - Knowledge Directions*, pages 157–189, New Delhi,India, 2003. Springer (India) Private Ltd.

[Dieng-kuntz and Matta, 2001] Rose Dieng-kuntz and Nada Matta. *Case-Based Reasoning*. Springer, 2001.

[Khemani *et al.*, 2002] Deepak Khemani, Radhika B. Selvamani, and Anand Rabi Dhar S M. Michael. Infofrax: Cbr in fused cast refractory manufacture. In *In Advances in Case Based Reasoning 6th European Conference EC-CBR06*, Aberdeen, Scotland, Sep 2002. Springer Verlag.

[Kolodner, 1993] Kolodner. *Case-Based Reasoning*. Morgan Kaufmann, SanMateo,California, 1993.

[Price C. and A., 1997] Ratcliffe M B. Price C., Peglar I S. and McManus A. From trouble shooting to process design: Closing the manufacturing loop. In *In proceedings of the 2nd International Conference ICCBR-97*. Lecture Notes in Computer Science, 1266, Springers Verlag, 1997.

[Quinlan, 1992] J. Ross Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufmann, 1992.

[Richter, 1995] M.M. Richter. The knowledge contained in similarity measures. In *Invited talk on ICCBR 95*, 1995.

[Selvamani and Khemani, 2005] B.Radhika Selvamani and Deepak Khemani. Decision tree induction with cbr. In *In proceedings of the First International Conference on Pattern Recognition and Machine Intelligence*, pages 786–791. Springer Verlag, December 2005.

[Watson, 2003] Ian Watson. *Applying Knowledge Management Techniques for Building Corporate Memories*. Morgan Kaufmann, San Fransisco, 2003.