Ontology Evolution for Customer Services

Tho T. Quan[1] and Thai D. Nguyen[2]
[1]Faculty of Computer Science and Engineering, Hochiminh City University of Technology
Hochiminh City
Vietnam

qttho@cse.hcmut.edu.vn
[2]Department of Information Technology, Nguyen Tat Thanh College
Hochiminh City
Vietnam

ndthai@ntt.edu.vn

Abstract

Customer service support has become an integral part of many multinational manufacturing companies which produce insertion and surface mount machines in the electronics industry. With the recent advancement of the Semantic Web, many attempts have been made to provide customer support over the Semantic Web environment.

Ontology is the major factor to represent knowledge on the Semantic Web. Therefore, a specific ontology so-called Machine Ontology is required to render customer service support with precise semantics. Machine Ontology can be generated either manually by human experts or automatically by intelligent programs. Even though manual ontology generation is highly accurate, it is tedious, time-consuming and requires high cost in terms of man power. On the other hand, automatic generation of ontology can provide high level of knowledge details and be effective when dealing with large-scaled dataset. However, automatically generated ontology may be not very semantically correct.

In order to tackle this problem, this paper proposes an ontology evolution technique that can enhance manual ontology with additional in-depth knowledge previously discovered in an automatic manner. The ontology evolution technique is based on the concept of ontology integration. The proposed technique has been applied to evolve Machine Ontology which is used to support customer service for industry manufacturers in Singapore. Some experimental results are also presented.

Acknowledgment

The authors wish to acknowledge the helpful support from Nguyen Tat Thanh College, Hochiminh City, Vietnam for the work presented in this paper.

1 Introduction

Currently, information available on the Web has been designed for human to understand. Programs can be written to process, analyze and index web pages to help us to process the information. However, due to the lack of machine-readable structure and knowledge representation in web documents, programs are unable to comprehend web page contents precisely, and hence semantic information from web documents cannot be extracted. The Semantic Web is therefore proposed as an extension to the current Web, in which information is given well-defined meaning, better enabling computers and people to work in cooperation (Berners-Lee *et. al.* 2001).

Ontology is used to represent knowledge on the Semantic Web. Generally, ontology is a conceptualization of a domain into a human understandable, but machine-readable format consisting of entities, attributes, relationships and axioms (Guarino and Giaretta 1995). As such, programs can use the knowledge from the Semantic Web for processing information in an intelligent manner.

Recently, the term ontology engineering has been used to imply ontology-related research in computer science (Mizoguchi and Ikeda 1996). The current issues on ontology engineering include ontology generation, ontology mapping (Maedche *et. al.* 2002, Omelayenko 2002), ontology integration (Gangemi *et. al.* 2002, Noy and Musen 2002, Stumme Maedche 2001), ontology versioning (Klein *et. al.* 2002) and ontology evolution. Particularly, when ontologies currently are more and more deployed in applications, the issue on ontology evolution techniques is attracting much attentions since it can help significantly ontology engineers to capture and reflect knowledge more precisely in certain domains (Stojanovic and Motik 2002, Noy and Klein 2003).

Basically, ontology evolution, first termed by Klein *et al.* (Klein *et. al.*, 2002), is a process which adapts the contents of a pre-defined ontology used in practical applications based on the environment in which the applications are deployed. In recent research, there are many techniques proposed for ontology evolution. Inspired from natural evolutions, biological concepts are suggested to be adopted and adapted for ontology evolution (O'Brien 2006). Another remarkable approach is using conceptual graphs combined with ontology editor tool such as Protégé (Blundell and Pettifer 2004). However, currently there are not much good practical

results reported from research on ontology evolution, due to the following reasons:

- Most of proposed techniques on ontology evolution heavily rely on manual methods. Thus, ontology evolution become a tedious and complex task, especially when representing large-scaled and in-depth domain knowledge.

- Generally, ontology is not formally defined before the ontology evolution process is performed. Thus, it is difficult to conduct a mathematical model which can help to automate ontology evolution.

In previous research (Quan *et. al.* 2006), we have proposed an approach for automatic ontology generation, which is based on the Formal Concept Analysis (FCA) theory (Ganter and Wille 1999). The ontology constructed by our approach is formally defined as a mathematical model employing FCA-based concepts for knowledge representation. The ontology can also present uncertainty information if necessary. Therefore, it provides a solid theoretical foundation for further study of ontology evolution.

In this paper, we propose an approach for ontology evolution. The ontology which is supposed to be evolved could be preliminarily generated either manually or automatically using our technique as well as other automatic ontology generation techniques. In order to evolve the ontology, we first analyze the in-depth knowledge of the domain of the application deploying the ontology. Then, we integrate the analyzed domain knowledge with the preliminary ontology. In addition, in this paper we also particularly discuss applying our proposed ontology evolution technique on a so-called Machine Ontology to suppor customer service in industry manufacturers.

The rest of the paper is organized as follows. In Section 2 we discuss Machine Ontology for customer service. Section 3 presents our proposed ontology evolution technique. Section 4 evaluates the performance based on some experiments conducted. Section 5 remarks some conclusion. The paper ends with references presented in Section 6.

2 Machine Ontology for Customer Service

Customer service support (Hui and Jha 2000), which renders helps while necessary for customers, has become an integral part of many multinational manufacturing companies which produce insertion and surface mount machines in the electronics industry. A customer service department is usually setup to provide prompt responses to service requests from customers, which may be located worldwide. The customer services of faulty machines are typically supported by a help-desk of the customer service department via telephone calls. This method relies heavily on the service engineers' past experience of similar reported problems, which is stored in a customer service database. Thus, a pool of expert service engineers is required in order to support such online customer services. With the advancement of the Internet, many manufacturing companies have moved their customer service support online over the Web with web-based help-desk applications (Hui *et. al.* 2001). Suggested remedial actions to the reported faults can be generated automatically from a customer service database or through the interaction between a user and service engineer in which the Web is used as a medium for communication.

Recently, the rapid development of the Semantic Web and Semantic Web Services (Austin *et. al.* 2002) has prompted us to consider supporting online machine services over the Semantic Web, which helps users to deal with problems occuring on the purchased machines. In a typical insertion and surface mount machine, there exist many machine models. Thus, a customer usually possesses different types of models and machines for manufacturing purposes. In such cases, when a machine fault problem occurs, the customer might need to obtain service support from different manufacturers. In conventional web-based service support systems, the customer is required to access different web sites, and retrieve different service support suggestions in order to rectify the machine fault problem. This is troublesome and cumbersome, and there is also the integration problem to be tackled.

As previously discussed, ontology is adopted as a standard for knowledge representation in the Semantic Web. Programs can use the knowledge from the Semantic Web for processing information in a semantic manner. As such, the Semantic Web enables machine service knowledge from different machines or models produced by different manufacturers to be shared and integrated. Moreover, the generated machine ontology can also be used to provide an interpretation on the common faults that have occurred for a certain machine model from a concept hierarchy. Thus, supporting machine services utilizying Semantic Web technologies will likely improve customer satisfaction in terms of reducing machine down time as well as increasing productivity.

This vision cannot become reality without a specific ontology known as *Machine Ontology* capturing machine knowledge in a computer-understandable manner. Machine Ontology can be generated either manually by experts or automatically by computer programs. Figure 1 presents an example Machine Ontology that is generated manually by experts. In this ontology, the experts have categorized the machine faults into a concept hierarchy.

Basically, this kind of ontology, so-called *Preliminary Machine Ontology*, is highly precise when reflecting domain knowledge, however since manual generation of ontology is generally tedious, the preliminary ontology is not always of sufficient details when deployed in practical applications.

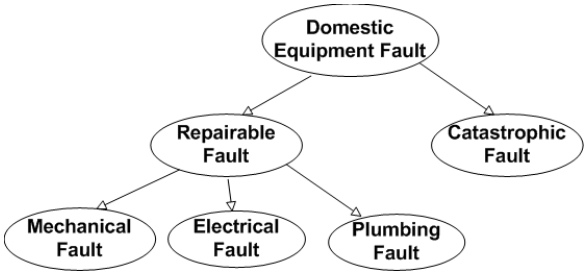

Figure 1: An example Preliminary Machine Ontology

Meanwhile, the initial knowledge stored in the ontology can be further processed using data mining to discover additional hidden information in the data. The discovered data can also be converted into ontological formalism to support more efficient retrieval tasks. Figure 2 illustratively depicts this kind of ontology, known as *In-depth Machine Ontology*. In this ontology, clustering technique is adopted to perform clustering on the dataset of machine faults and then a hierarchy is constructed based on the clusters generated.

The in-depth ontology should reflect specific knowledge of the domain. However, since the in-depth knowledge is quite extensive and profound, it lacks of a clear and precise structure that enables users with basic domain knowledge to easily follow, understand and deploy the knowledge.

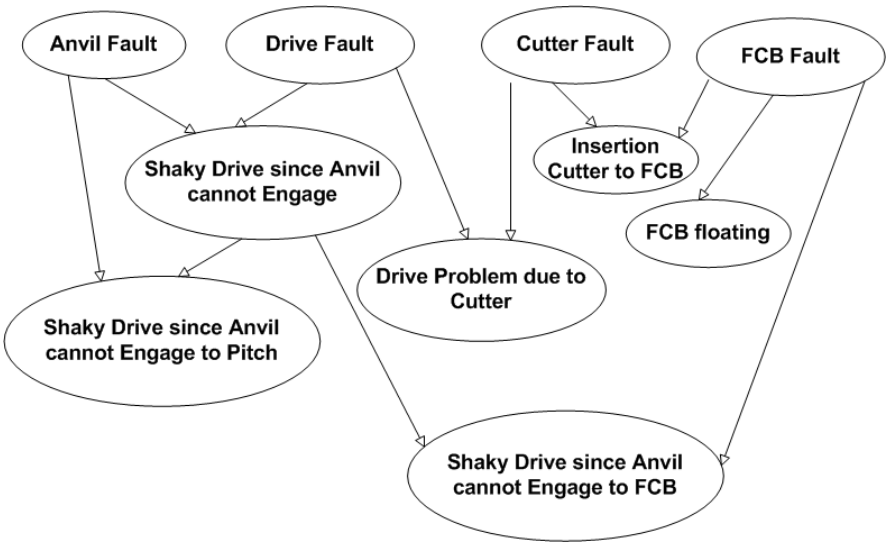

Figure 2: An example In-depth Machine Ontology

In this paper, we propose to use the in-depth ontology to make the preliminary ontology evolved accordingly into a final evolved ontology. The evolved ontology could be considered as an enhanced version of the preliminary ontology as it still preserves precisely the concept hierarchy inherited from the preliminary ontology but enriched by additional specific knowledge derived from the in-depth ontology. The evolved ontology therefore reflects the domain knowledge in a far deeper level but still maintaining the structural accuracy predefined by experts.

3 Ontology Evolution

In order to process ontology evolution, we perform ontology integration from the preliminary ontology with the in-depth ontology. In our proposed approach, ontology integration is considered as a process that integrates concepts in a preliminary ontology with those of a in-depth ontology in order to generate a new ontology. Thus, the preliminary ontology is extended with new concepts and relations derived from the in-depth ontology. Ontology integration consists of two sub-steps: (1) Hierarchical Concept Integration and (2) Consistency Checking and Refinemen

3.1 Hierarchical Concept Integration

Basically, an ontology includes classes, or *concepts*, and relations between these concepts; and the instances of the classes. Formally, an ontology can be defined as follows.

Definition 1 (Ontology). An ontology O consists of 4 elements (C,A^C,R,X), where C represents a set of concepts; A^C represents a collection of attributes sets, one for each concept; $R = (R_T, R_N)$ represents a set of relationships, which consists of 2 elements: R_N is a set of non-taxonomy relationships and R_T is a set of taxonomy relationships. Each concept c_i in C represents a set of objects, or instances, of the same kind. Each object o_{ij} of a concept c_i can be described by a set of attributes values denoted by $A^C(c_i)$. Each relationship $r_i(c_p,c_q)$ in R represents a binary association between concepts c_p and c_q, and the instances of such a relationship are pairs of (c_p,c_q) concept objects.

To integrate a Preliminary Ontology O_P and an In-depth Ontology O_I , each concept c in O_I will be mapped as concept c' in O_P, in which c' can be either an old concept already existing in O_P or a new concept of O_p. In case c' is a new concept in O_P, corresponding hierarchical relations will be accordingly constructed to make c' appropriate sub-concepts or super-concept of existing concepts in O_P.

Thus, in order to integrate concepts in two ontologies reasonably, it is necessary to develop a method to measure the *semantic similarities* of ontological concepts. In literature, there are two major trends of measuring semantic similarities for ontologies. The first approach is based on the structure of ontological concepts (Li *et. al.* 2003, Rada *et. al.* 1989), the second takes into account the internal information of the concepts for similarity measurement (Jiang 1997, Lin 1998, Resnik 1995).

In the context of this paper, we are about to integrate a preliminary ontology with an in-depth one. Thus, the conceptual structure of the two ontologies are quite distant and therefore not much informative for calculating concepts similarities. Hence, we will analyze and make use of the internal information (or *attributes*) of ontological concepts to measure their similarity. First, we will represent attributes of a certain ontological concept as a *fuzzy set*, known as *fuzzy representation* of the concept. (Note that to represent the concept precisely, fuzzy information is needed to reflect some uncertainty information that may occur in some concept attributes). Then we make use fuzzy logic to perform similarity measurement in the generated fuzzy sets. In addition, since a concept can be mapped not only as the same concept but also as a subconcept or superconcept of another concept, we also use fuzzy logic to evaluate the *subsethood* between the concepts.

The above discussion is formally represented as follows.

Definition 2 (Fuzzy Representation of Ontological Concepts). Given a fuzzy ontology (C,A^C,R,X), each concept $c \in C$ can be represented as a fuzzy set, or *fuzzy representation*, *FR(C)* as

$$FR(C) = \{A_1(\mu_1), A_2(\mu_2), ..., A_m(\mu_m)\}$$

where $\{A_1, A_2, \ldots, A_m\}$ is the set of attributes in A^C and μ_i is the fuzzy membership A_i, calculated as follows:

$$\mu_i = \frac{n_i}{n}$$

where n_i is number of objects of C that have certain value on A_i and n is the number of the objects of C. It is easy to observe that if C have no uncertainty information, μ_i is 1 for any i.

Definition 3 (Ontological Concept Similarity). The similarity between two ontological concepts c_i and c_j is evaluated as

$$E(c_i, c_j) = \left| \frac{FR(c_i) \cap FR(c_j)}{FR(c_i) \cup FR(c_j)} \right|$$

Definition 4 (Ontological Concept Subsethood). The subsethood between two ontological concepts c_i and c_j is evaluated as

$$subsethood(c_i, c_j) = \left| \frac{FR(c_i) \cap FR(c_j)}{FR(c_j)} \right|$$

Thus, the concept similarity and concept subsethood defined above can significantly help to determine whether or not a concept in a In-depth Ontology should be mapped as a subconcept of or unified with an existing concept in a Preliminary Ontology. It is explained in more details in the following example.

Let's consider a Preliminary Ontology P_I as presented in Figure 3. P_I has three concepts, namely *Mechanical Fault*, *Anvil Fault* and *Drive Fault*. Basically, the concept *Mechanical Fault* describes all of possible mechanical problems on a certain machine model; whereas the concepts *Anvil Fault* and *Drive Fault* respectively concern faults specifically occurring in the anvil and drive components of the machine.

This preliminary ontology is supposedly constructed manually by an ontology engineer. When defining these ontological concepts, the ontology engeneer may as well define some concept attributes, which are some keywords related to the concepts extracted from the user manual of the model. Note that we assume that ontologies involved in our technique conform with concept definition in FCA theory. In this ontology, the *Mechanical Fault* concept is the most general concept. Thus it has theoretically no attributes, as stipulated by FCA theory. Meanwhile, the concepts *Anvil Fault* and *Drive Fault* are defined respectively by the set of attributes {*shaped, hammering, chiseling, forging, steel, plate*} and {*operate, control, force, repel, strike, throwing*}. For simplification purpose, we suppose that the membership values of all attributes are 1 (meaning all of them are not uncertain attributes). For convenience, these concepts are hereinafter referred to as *A*, *B* and *C* as noted in Figure 3.

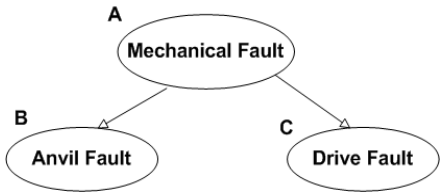

Figure 3: The Preliminary Ontology P_I

Suppose that we intend to integrate P_I with an ontology named In-depth Ontology E_I as depicted in Figure 4. E_I is constructed from real record of fault occuring in the model, collected from users. Basically, in E_I, the fault records are organized as a hierarchy of concepts, each of which is represented by a corresponding set of attributes as shown in Figure 4. The attributes here are once again keywords extracted from the reports of the corresponding fault cases. These concepts are also hereinafter referred to as *D*, *E*, *F*, *G* and *H*.

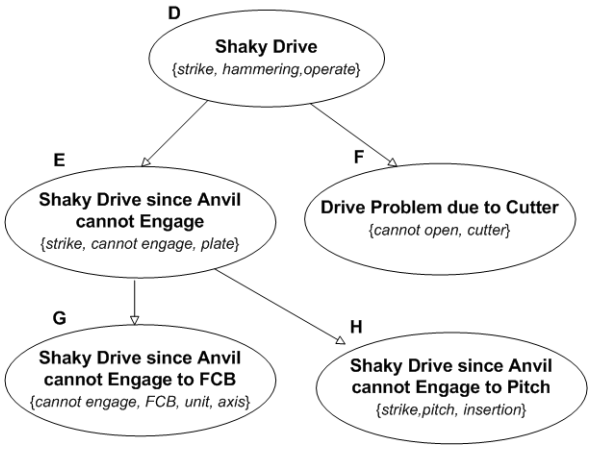

Figure 4: The In-depth Ontology E_I

The ontology integration process, subsequently, is divided into two sub-steps, namely Subsethood-based Integration and Similartity-based Integration.

3.1.1 Subsethood-based Integration

In Subsethood-based Integration process, each concept in E_I will first be mapped as either a subconcept or a superconcept of appropriate concepts in P_I. The mapping is determined by evaluating the subsethood between concepts. For example, consider concepts *A* in P_I and *D* in E_I. We have *subsethood*(*C*,*D*) = 0.67. Since this subsethood value is considerably high, *D* then is mapped as a subconcept of *C*. In other hand, *subsethood*(*C*,*G*) = 0.25 and *subsethood*(*G*,*C*) = 0.125. Since *subsethood*(*G*,*C*) is too low, we do not consider mapping *C* as a subconcept of *G*. Instead, *G* can still be considered as a subconcept of *C*, due to the acceptably high value of the corresponding subsethood.

Thus, after fully mapped, the concepts on E_I will be initially integrated into P_I based on the corresponding subsethoods evaluated. As a result, an initial integration ontology is generated as illustrated in Figure 5. Note that in the initial ontology, *D* is mapped as concurrenly a subconcept as well as a superconcept of *C*. It is because both values of *subsethood*(*C*,*D*) and *subsethood*(*C*,*D*) are quite high (0.67 and 0.33 respectively). The ontology will be refined to solve this problem in the later steps.

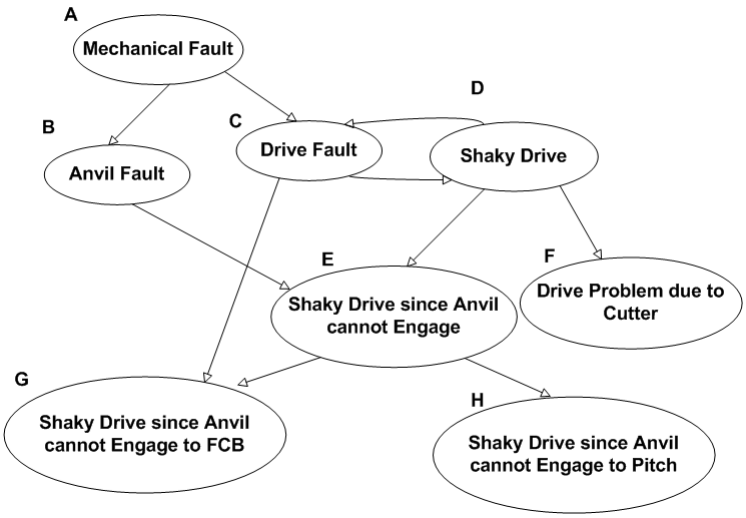

Figure 5: Initial integration ontology based on subsethood.

3.1.2 Similarity-based Integration

The next step to be performed on the initial ontology is to evaluate the similarities between the concepts. Based on the similarities evaluated, we unify the concepts on the initial ontology if their similarities are considerably high. For example, on the initial ontology given in Figure 5, we have *E*(*B*,*E*) = 0.33. This similarity is considerably high, urging us to consider unifying these two concepts together as a new concept *BE*. As a result, the final integration ontology is generated as shown in Figure 6.

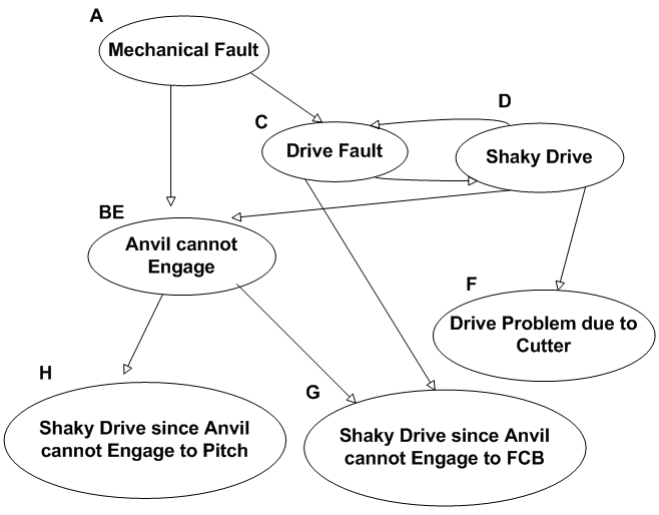

Figure 6: Integration ontology of P_I and E_I

3.1.3 Ontology Integration Algorithm

So far, we have introduced the key ideas for integrating ontologies, based on the definitions of ontological concept subsethoods and similarities. The above discussion on ontology integration can be formally presented as Algorithm 1 given in Figure 7.

Algorithm 1: Ontology Integration

Input: Preliminary Ontology O_P and In-depth Ontology O_I
Output: Evolved ontology O_E
Process:

```
1:  O_E ← O_P
2:  for each concept c concept of O_I do
3:  map c as a new concept c_N of O_E
4:  for each concept c_E concept of O_E do
                  if subsethood(c_N, c_E) > T_C then
                        make C_E subconcept of C_N
                  endif
                  if subsethood(c_E, c_N) > T_C then
                        make C_E superconcept of C_N
                  endif
5:  endfor
6:  find concept c_cand of O_E that is most similar to c_N
7:  if E(c_cand, C_N) > T_C then
                  make every subconcept of C_N subconcept of c_and
                  make every superconcept of C_N superconcept of c_and
                  remove c_N from O_E
8:  endif
9:  endfor
```

Figure 7. Ontology integration algorithm

3.2 Consistency Checking and Refinement

As previously discussed, the evolved ontology is generated mathematically without human control. Thus, there may be some inconsistency occuring in terms of concept relations. It is because there are some relations are accepted by a our mathematical model, but they do not make sense when concerned in the practical situation. For example, in the integration ontology given in Figure 5, concept *D* is at the same time subconcept and superconcept of concept *C*. Obviously, it makes the ontology unlogical and not natural. Such inconsistency should be refined.

In this research, when checking for inconsistency in the evolved ontology, we only focus on the inconsistency on hierarchical relations of concepts, defined as follows.

Definition 5 (Hierarchical Relation Inconsistency). Given an ontology *O*, a *hierarchical relation inconsistency* will occur in two concepts C_1 and C_2 of *O* if C_1 is concurrently a superclass and a subclass of C_2.

When a hierarchical relation inconsistency occurs in two concepts C_1 and C_2, we will refine the ontology using the information on subsethood of the concepts. That is, if *subsethood*(C_1,C_2) > *subsethood*(C_2,C_1), the taxonomy relation which specifies that C_1 is superclass of C_2 will be removed from the ontology (meaning that C_1 now is only a subclass and no longer superclass of C_2) and vice verca. For example, in the integration ontology mentioned above, we have *subsethood*(*C*,*D*) > *subsethood*(*D*,*C*). Therefore, we refine the ontology to make *D* no longer superconcep of *C*. As a result, we have the refined ontology as presented in Figure 8.

After fully refined, the ontology evolution process is considered complete. As shown in Figure 8, the Preliminary Ontology is now evolved with new concepts raised from In-depth Ontology. Moreover, the initial relations between original concepts have also been refined to adapt with new knowledge obtained.

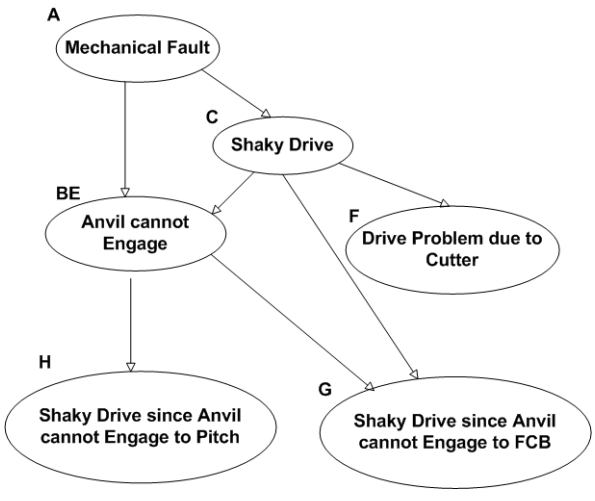

Figure 8. Final evolved ontology

4 Performance Evaluation

In order to evaluate the peformance of the proposed technique, we have used a real *Customer Service Database* for ontology generation and evolution. The

Customer Service Database records fault occuring on products of a multinational corporation in Singapore that manufactures insertion and surface mount machines in the electronics industry. In the database, fault records (or reports) are currently defined and stored in the customer service database to keep track of all reported machine problems and remedial actions. Each service record consists of customer account information and service details.

Customer service details contain two types of information: *fault-condition* and *checkpoint information*. Fault-condition contains the service engineer's description of the machine fault. Checkpoint information indicates the suggested actions or services to be carried out to repair the machine into normal condition based on the occurred fault-condition given by the customer. Checkpoint information contains checkpoint group name, and checkpoint description with priority and an optional help file. The checkpoint group name is used to specify a list of checkpoints defined under the group. Each checkpoint is associated with a priority, which determines the sequence in which it can be exercised, and a help file that gives visual details on how to carry out the checkpoint. An example of fault-condition and its checkpoint information for a service record is given in Figure 9.

There are over 70,000 service records in the customer service database. Since each of the fault-conditions has several checkpoints, there are over 50,000 checkpoints. In addition, information on over 4,000 employees, 500 customers, 300 different machine models and 10,000 sales transactions are also stored.

Fault-condition	3008 PCB CARRY MISS ERROR. PCB WAS NOT TRANSFERRED BY THE CARRIER DURING LOADING BUT STAYED AT THE DETECTION POSITION OF PCB DETECTION SENSOR 2.	
Checkpoint group**:** AVF_CHK007		
Priority	Checkpoint description	Help file
1	CONFIRM WHETHER THE CARRY GUIDE PINS ARE IN LINE WITH PCB.	AVF_CHK007-1.GIF
2	CONFIRM WHETHER THE PCB IS IN CORRECT DIRECTION.	AVF_CHK007-2.GIF
3	CONFIRM THE POSITION OF THE GUIDE LOWER LIMIT SENSOR. (I/O 0165)	AVF_CHK007-3.GIF
4	CONFIRM THE TIMING FOR PCB 2 DETECT SENSOR.	AVF_CHK007-4.GIF
5	CONFIRM THE TIMING FOR THE CARRIER START TIMING.	AVF_CHK007-5.GIF

Figure 9: Customer Service Database

Based on the relational schema of the Customer Service Database, we define the Preliminary Machine Ontology, as illustratively depicted in Figure 1. Subsequently, clustering techniques that can be used to cluster machine fault conditions into groups based on their similarities are traditionally used for this purpose.

In this research, we have applied one of the most effective clustering techniques known as Kohonen Self-Organizing Map (KSOM) Neural Networkfor knowledge discovery. From the clusters generated, we automatically generated the In-depth Machine Ontology, as seen in Figure 2. The ontology generation technique is based on Formal Concept Analysis (FCA) technique.

From the Preliminary Machine Ontology and In-depth Machine Ontology, we generate the Evolved Machine Ontology, as illustrated in Figure 10. The purpose of the evolved ontology is to support customer services of past common fault information and current fault diagnosis. Therefore, an experiment has been conducted to evaluate the system performance of information retrieval.

There are two testing datasets conducted for retrieval accuracy evaluation. The first dataset, denoted as *DS1*, is a set of 15,850 fault-conditions, used as input strings for retrieval. As the fault descriptions in the fisrt set were manually created by experts, the second set, denoted as *DS2*, consists of 50 fault descriptions formed from non-expert users. It is used to test the performance when the input is less technically precise.

KSOM is an unsupervised learning technique, therefore it can automatically discover hidden knowledge in data without using prior knowledge given by human experts. For experimental comparison purpose, a supervised neural network, Learning Vector Quantization Version 3 (or LVQ3), is also adopted in the experiment. In addition, *k*-nearest neighbor (kNN) technique has been typically used for retrieval purpose in traditional customer service system. Therefore, apart of two mentioned neural networks, we also made use of kNN for retrieval evaluation. There are two popular variations of kNN techniques were adopted. The first variation, denoted as kNN1, bases on vector's normalized Euclidean distance to find the fault-conditions closest to the input string. The second, denoted as kNN2, makes use of fuzzy-trigram technique to do so. Then, we evaluate the performance of adopted techniques using two criteria: speed and accuracy.

4.1 Retrieval Speed Evaluation

We evaluate the speed performance based on average on-line retrieval. For KSOM and LVQ3 neural networks, training time was also measured and compared. As a unsupervised learning technique, KSOM can automatically make use the data in the Customer Service Database for training. However, LVQ3 needs human interpretation to determine target fault-condition corresponding to each given training input.

The dataset *DS1* was used for speed evaluation. The statistics information of training and online-average retrieval time of four mentioned techniques is given in Table 1. It states that even though kNN techniques do not

need to be trained before performing retrieval, their performance was outperformed by neural networks techniques in terms of speed. As compared to LVQ3, KSOM requires longer time for training, but it performs more efficiently in terms of on-line retrieval speed. Since the training process can be carried out in off-line mode, neural networks, particularly KSOM, are more suitable for an on-line retrieval system than the typical kNN technique.

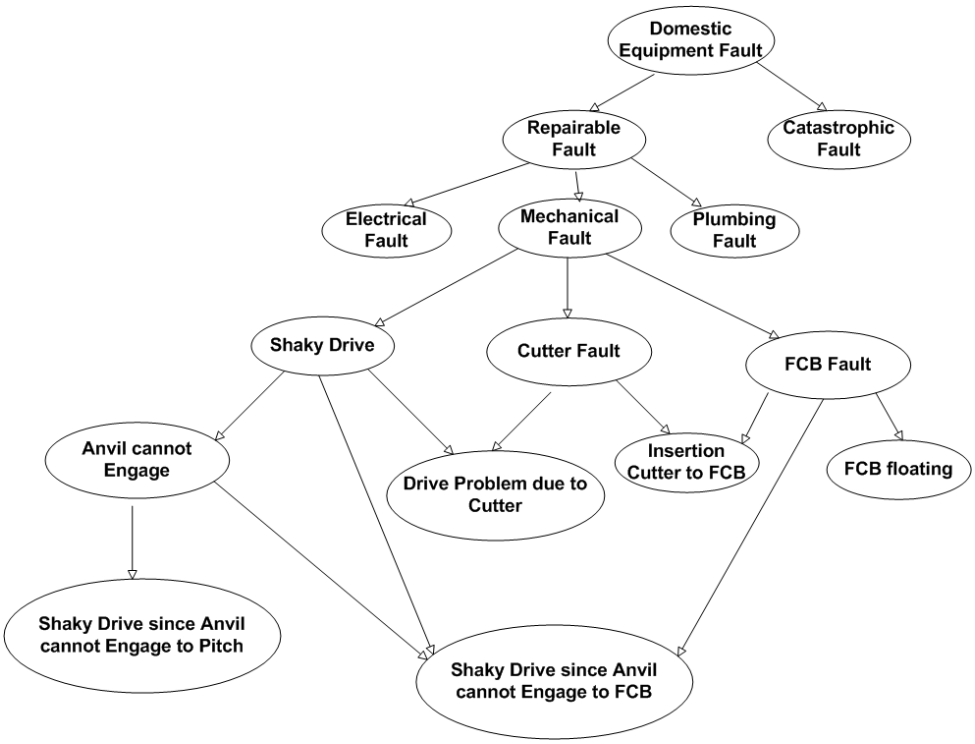

Figure 10: Final evolved Machine Ontology

Description	Time			
	LVQ3	**KSOM**	**kNN1**	**kNN2**
Training	96 min 44 sec	264 min 35 sec	N/A	N/A
Average on-line retrieval	1.9 sec	0.8 sec	15.3 sec	16.7 sec

Table 1: Retrieval Speed Evaluation

Retrieval Technique	Retrieval Accuracy	
	Based on *DS1*	**Based on *DS2***
kNN1	81.4%	72%
kNN2	77.6%	76%
LVQ3	93.2%	88%
KSOM	90.3%	86%

Table 2: Retrieval Accuracy Evaluation

4.2 Retrieval Accuracy Evaluation

Retrieval accuracy was evaluated based on the accuracy of ault-conditions retrieved. The way to evaluate retrieval result of KSOM has been discussed earlier. The retrieval accuracy of LVQ3 and kNN techniques could be determined directly through the correctness of fault-conditions retrieved. Table 2 gives the accuracy of retrieval measured on two datasets *DS1* and *DS2*. As can be seen in the table, the accuracies of two neural networks KSOM and LVQ3 were better than those of kNN technique variations. The retrieval accuracy on the dataset *DS2* set was lower than on *DS1*. It is expected due to the fact that technical terms might be not properly used by non-expert users when conducting fault-conditions in *DS2*. In both data sets, LVQ3 achieved better performance than KSOM. It is also expected since a supervised learning technique often obtains better accuracy than an equivalent unsupervised one when applied in a same domain. However, since LVQ3 always needs human interpretation for training, it may encounter problem when dealing with large dataset.

5 Conclusion

This paper presents an ontology evolution technique. This technique allows an initial preliminary ontology to be evolved with new information that is generated automatically from large datasets. Thus, whereas the evolved ontology still maintains the original structural correctness, it is enriched reasonably with new in-depth knowledge which is generally hard for human being to extract manually.

In particular, the proposed ontology evolution technique has been applied to evolve Machine Ontology generated from the real Customer Service Database which records experienced faults occuring in industry manufacturers in Singapore. The evolved Machine Ontology is therefore potential to support customer service for e-commerce in the coming Semantic Web environment.

6 References

Austin D., Barbin A., Ferris C. and Garg S. (2002), Web Services Architecture Requirements. Available at: <http://www.w3c.org/TR/wsa-reqs>, 2002.

Berners-Lee, T., & Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*. Retrieved from http://www.sciam.com/2001/0501issue/0501berners-lee.html, 2001.

Blundell B. and Pettifer S. (2004): Graph Visualization to Aid Ontology Evolution in Protégé, In *Proceedings of 7th International Protégé Conference*, July 2004.

Gangemi A., Fisseha F., Pettman I., Pisanelli D., Taconet M, and Keizer J. (2002): A Formal Ontological Framework for Semantic Interoperability in the Fishery Domain, In *Proceedings of ECAI02 Workshop on Semantic Interoperability*, 2002.

Ganter B. and Wille R. (1999): *Formal Concept Analysis: Mathematical Foundations.* Springer, Berlin - Heidelberg, 1999.

Guarino N. and Giaretta. P. (1995) Ontologies and Knowledge Bases: Towards a Terminological Clarification. Toward Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing. Amsterdam, Holland: IOS Press, 1995.

Hui S.C., Fong A.C.M. and Jha G. (2001): A Web-based Intelligent Fault Diagnosis System for Customer Service Support, *Engineering Applications of Artificial Intelligence*, **14**: 537-548, 2001.

Hui S.C. and Jha G. (2000): Data Mining for Customer Service Support, *Information & Management*, **38**:1-13, 2000.

Jiang, J.J, and Conrath, D.W (1997). Semantic Similarity Based on Corpus Statistics and Lexical Ontology. *In Proceedings on International Conference on Research in Computational Linguistics*, 19–33,1997.

Klein M., Fensel D., Kiryakov A., and Ognyanov D. (2002): Ontology Versioning and Change Detection on the Web, In *Proceedings of the OntoWeb-SIG3 Workshop at the 13th International Conference on Knowledge Engineering and Knowledge Management*, October 2002.

Li, Y., Bandar, Z. A. and McLean D. (2003): An Approach for Measuring Semantic Similarity between Words Using Multiple Information Sources. *IEEE Transactions on Knowledge and Data Engineering*, **15**(4): 871-882, 2003.

Lin, D (1998). An Information-theoretic Definition of Dimilarity. In *Proceedings of the Int'l Conference on Machine Learning*, 1998.

Maedche A., Motik B., Silva N., and Volz. R. (2002): MAFRA - An Ontology Mapping Framework in the Semantic Web, In *Proceedings of the ECAI Workshop on Know-ledge Transformation, Lyon, France*, 2002.

Mizoguchi R. and Ikeda M. (1996): Towards Ontology Engineering, *technical report*, Osaka University, 1996.

Noy N. and Musen M. (2002): PROMPTDIFF: A Fixed-Point Algorithm for Comparing Ontology Versions, in *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI'02)*, (Alberta, Canada), 2002.

Noy N. and Klein M. (2003): Ontology Evolution: Not the same as schema evolution, *Knowledge Information Systems*, **5**, 2003.

Omelayenko (2002): RDTF: a Mapping Meta-ontology for Business Integration, In *Proceedings of the Workshop on Knowledge Conference on Artificial Intelligence for the Semantic Web at the 15th European Conference on Artificial Intelligence(KTSW2002)*, Lyon, France, 2002.

O'Brien P. (2006): Modeling Intelligent Ontology Evolution Using Biological Evolutionary Processes, *Engineering of Intelligent Systems, IEEE International Conference*, 2006.

Quan T.T., Hui S.C., Fong A.C.M. and Cao H.T. (2006) : Automatic Fuzzy Ontology Generation for Semantic Web, *IEEE Transactions on Knowledge and Data Engineering*, **18**(6): 842- 856, 2006.

Rada, R., Mili, H. Bicknell, E. and Blettner, M (1989). Development and Application of a Metric on Semantic Net. *IEEE Transactions on Systems, Man and Cybernetics*, **19**(1):17-30, 1989.

Resnik, P. (1995): Using Information Content to Evaluate Semantic Similarity in Ontology. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*,448–453,1995.

Stojanovic L. and Motik B. (2002): Ontology Evolution within Ontology Editor, In *Conference on the Evaluation of Ontology-based Tools*, September 2002.

Stumme G. and Maedche A. (2001): FCA-merge: Bottom-up Merging of Ontologies, In *Proceedings of the 17th International Conference on Artificial Intelligence (IJCAI '01)*, (USA): 225-230, 2001.