Authoring cases from Free-Text Maintenance Data

Yang, Chunsheng; Orchard, Robert; Farley, Benoît; Zaluski, Marvin

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC-CNRC

*Authoring cases from Free-Text Maintenance Data **

Yang, C., Orchard, R., Farley, B., Zaluski, M.
July 2003

* published in Proceeding of IAPR International Conference on Machine Learning and Data Mining (MLDM 2003). Leipzig, Germany. July 5-7, 2003. NRC 45813.

Canada

Authoring Cases from Free-Text Maintenance Data

Chunsheng Yang[1], Robert Orchard[1], Benoit Farley[1], and Maivin Zaluski[1]

[1] National Research Council, Ottawa, Ontario, Canada
{Chunsheng.Yang, Bob.Orchard, Benoit.Farley, Marvin.Zaluski}@nrc.ca

Abstract

Automatically authoring or acquiring cases in the case-based reasoning (CBR) systems is recognized as a bottleneck issue that can determine whether a CBR system will be successful or not. In order to reduce human effort required for authoring the cases, we propose a framework for authoring the case from the unstructured, free-text, historic maintenance data by applying natural language processing technology. This paper provides an overview of the proposed framework, and outlines its implementation, an automated case creation system for the Integrated Diagnostic System. Some experimental results for testing the framework are also presented.

Keywords: cased-based reasoning, case creation, case base management, natural language processing.

1. Introduction

The Integrated Diagnostic System (IDS), which was developed at the National Research Council of Canada, is an applied artificial intelligent system [2] that supports the decision-making process in aircraft fleet maintenance. IDS integrates two kinds of the reasoning techniques: rule-based reasoning and case-based reasoning. The rule-based reasoner monitors the messages transmitted from the aircraft to the information monitoring system on the ground. The messages are either malfunction reports from the sensors of an aircraft (failure [FLR] or warning [WRN] messages) or digital messages typed on the keyboard by the pilot (SNAG[1] or MSG messages). IDS clusters these messages into different Fault Event Objects (FEOs), which are regarded as potential problem symptoms. These symptoms trigger the firing of rules that alert maintenance technicians to situation that could have a significant impact on the aircraft's airworthiness. IDS also helps identify the appropriate parts of troubleshooting manual that are related to the symptoms. CBR [1] is then needed to help refine these solves by retrieving similar situations from the mechanic's experiences, which have been stored in a case base.

The case bases are different from the rule bases in principle. The rules reflect the relationship between condition and consequence in real-world problems; they can

[1] *A snag is a common term for an equipment problem in the aviition area. It is a record of the problem and the repair action.*

be designed based on system requirements and domain knowledge, or extracted from the technical documents such as the troubleshooting manual. The cases document the relation between problem symptoms and the fix applied by domain experts, and they accumulate the past experience for solving similar problems. The cases can't be created from technical documentation. They have to be authored from historic maintenance experience or by experienced domain experts.

One important piece of data is the snag message. A snag is a transcript of the hand-written notes describing a problem (reported by pilots, other crew or maintenance technicians) and the repair actions carried out to fix the problem. It is composed of well defined, fixed fields describing the date, the location, a unique snag identifier, etc. as well as unstructured free-text describing the problem symptoms, the pieces of equipment involved in the repair and the actions performed on them. It is possible for someone to create a potential case by combining the information in the snag message with information in the FEO database. To help the user to create cases from the historic snag database, we developed an off-line tool, SROV (Snag Ratification Object Validation)[1]. This tool allows the user to browse the snag database, clean up the contents of the snag message and convert the snag message into a potential case. However, it was still difficult for the user to create cases using the tool, because the problem description and repair action in the snag messages are described with unstructured free text. To extract useful information from such free-text messages requires significant human effort and domain knowledge.

In order to reduce the human effort, we propose a framework for authoring cases automatically from the unstructured free-text maintenance data by applying natural language processing (NLP) techniques [3][14]. In this paper, the proposed framework is presented in detail along with its implementation, an automated case creation system (ACCS) for IDS. Some experimental results for testing the effectiveness of the framework are also discussed.

The paper is organized as follows. Following this introduction is Section 2, Related Work; Section 3 is the proposed framework; Section 4 describes the technical implementation of ACCS; Section 5 presents some experimental results; and the final section discusses the conclusions.

2. Related Work

To date a great deal of research effort has been devoted to case base maintenance [4,5,6,7,9,10,12] in CBR systems. This research has focused on a number of crucial issues such as the case life cycle [5], the optimization of the case indices [12] and so on. Some of the earliest case base maintenance works [9,10] look at the development of maintenance strategies for deleting/adding cases from/to existing case bases. For example, in [9], a class of competence-guided deletion policies for estimating the competence of an individual case and deleting an incompetent case from a case base is presented. This technique has been further developed for adding a case to an existing case base [10]. Redundancy and inconsistency detection for case base management in CBR systems has also attracted a lot of attention from researchers [11]. In recent years, some new approaches based on automatic case base management strategies have been published. M.A. Ferrario

and B. Smyth [6], introduced a distributed maintenance strategy, called collaborative maintenance, which provides an intelligent framework to support long-term case collection and authoring. To automatically maintain the case base, L. Portinale et al [4] proposed a strategy, called LEF (Learning by Failure with Forgetting [13]), for automatic case base maintenance.

It is perhaps surprising that these works almost exclusively focus on maintaining case bases for runtime CBR systems and collecting cases from the on-line problem-solving procedures. Relatively little work has focused on automatically authoring cases at an earlier stage, using existing historic maintenance experience that can be collected from past maintenance operational data. In fact, a useful CBR system should provide the ability for a user to automatically author case bases from the recorded historic experience database at the initial stage and to automatically collect or author the cases at the on-line runtime stage. Therefore, the main contribution of this paper is to propose a useful framework for automatically authoring cases from the historic maintenance experience data by applying NLP techniques.

3. A Framework for Authomatically Authoring Cases

To describe the proposed framework, we use the following notations. Let c denote a *case* and CB denote a case base, then $CB \supseteq (c_1, c_2 ....., c_i, ..., c_n)$. A case $\boldsymbol{c}$ is defined as $c = ((p),(s),(m))$ where *(p), (s)* and *(m)* denote problem attributes (called symptoms), solution attributes to the problem and information for case base management respectively. *(m)* contains all attributes related to case base maintenance including redundancy, inconsistency, positive actions, and negative actions. *(p)* could be a single symptom or multiple symptoms, and *(s)* could be a single action or multiple actions for fixing the problem *(p)*. If SB and FB denote the historic snag maintenance database and the FEO database respectively, then $SB \supseteq (snag_1, snag_2, ...snag_k)$ and $FB \supseteq (f_1, f_2, ...f_l)$. Our task is to create CB from SB and FB. Therefore, the framework can automate this task by following five main processes:

- Preprocessing snag messages (*SB*),
- Identifying the symptoms(*(p)*) for the problems,
- Identifying the solution (*(s)*) for the problems,
- Creating a potential case (*c*),
- Maintaining the case base (*(m)*).

3.1 Preprocessing snag messages

The task of this process is to obtain clean snag messages $snag_i \subseteq SB$ from the raw snag messages. The raw snag messages like the one shown in Table 1, are processed to give messages in italics as shown in Table 2. The parse is simple since the various fields of the raw message are in a predetermined order and of the fixed size. We extract the date, the place where the fix was done, a unique snag identifier, etc, as well as unstructured free-text describing the problem symptoms and the repair actions. The free-text contains many unnecessary symbols or words. To deal with this, we filter the unnecessary characters (such as '#', '.', '*' and so on) and using a list of "poor single" words, we remove some words as well. The list of poor single words are constructed by analyzing a large set of snag messages

to see which ones were not helpful in matching the unstructured text FLR and WRN messages. For example, the free-text of the problem description obtained from the raw snag message, *RMA 27-93-2127 AVAIL. REPEAT E/W "F/CTL ELAC 1 FAULT" "ELAC 1 OR INPUT OF CAPT ROLL CTL SSTU 4CE1". R 7.* after processing, results in *RMA 27-93-2127 AVAIL REPEAT F/CTL ELAC 1 FAULT ELAC 1 INPUT CAPT ROLL CTL SSTU 4CE1,* as shown in Table 2.

Table 1: An example of the raw maintenance data record

ACFT_MI_SEC:UNNNNNNNNNNNNNNNNNNNNNNNYYYYYYYYYYYYYYYNYNNNNNNNNYYNN6615 437820001N**M1003286** 2312 2312ACA01058P28Q0CL6**YUL** ACA0646RT **RMA 27-93-2127 AVAIL. REPEAT E/W "F/CTL ELAC 1 FAU LT" "ELAC 1 OR INPUT OF CAPT ROLL CTL SSTU 4CE1". R 7.** I2000-09-23NNDEFN 0000000000000 0000000000000 0000000000000 00000000000000 40227AC 74577LNNS ORDER AC74577 **1998-01-22 14:07:00**6650 ACFT_MI_ACTN_SEC : INNNNNNNNNNNNNNNNNNNNNYYYYYYNN 615437820002000 6889450001Y **REPLACED CAPTAINS SIDE STICK AND TESTED AS PER AMM 27-92-41-501** 42000-09-2506.36.00FIX**YWG** 26525AC 26525NNNNNN 000000000000 AC26525 **1998-01-30 16:00:00**.898990 ACFT_PART_RMVL_SEC:NNNNNNNNNNNNNNNNNNNNNNN661543782000200068894500010001 001Y0000000010000NNNAC002FD 9W19XFEA 150000000042983622-9852-003 4V792 111AC26525 **1998-01-30 16:00:00**.89916023-80-0100 Y ACFT_PART_INST_SEC:NNNNNNNNNNNNNYNYYNYNN6615437820002000 688945000 100010001 Y0000000010000NN AC002EA 150000000042983 1467 AC26525 **1998-01-30 16:00:00**.89921023-80-0100 Y

Table 2: A clean snag message obtained from the Table 1

Event Date & Time	*1998-01-22 14:07:00*
Report Station	*YUL*
Snag Number	*M1003286*
Problem Description	*RMA 27-93-2127 AVAIL REPEAT F/CTL ELAC 1 FAULT ELAC 1 INPUT CAPT ROLL CTL SSTU 4CE1*
Fin Number	*222*
Repar Station	*YWG*
Repair Date	*1998-01-30 16:00:00*
Repair Action	*REPLACED CAPTAINS SIDE STICK AND TESTED AS PER AMM 27-92-41-501*

3.2 Identifying the symptoms

The task of the process, symptom identification, is to find *(P)* from the *SB and FB*. Identifying the symptoms for the problem is done using a free-text matching approach because the content of the diagnostic FLR and WRN messages is described in formal (predetermined) text while the problem description in the snag message is unstructured free text. To match such free text to the formal text of the diagnostic messages, we use an N-gram algorithm. N-gram matching refers to a fragment of N consecutive letters of a text phrase. For a given text phrase of length L, there are $L-N+1$ N-grams. Such a matching algorithm helps to reduce the impact of misspelling, abbreviations and acronyms. After considering the trade-off between the algorithm performance and matching accuracy, we selected N to be 3

(tri-gram matching). For example, in the tri-gram matching algorithm, the text word "***diagnose***" could be disassembled into 6 tri-grams: $\{dia, iag, agn, gno, nos, ose\}$. If a text phrase, "***diagnose"*** is matched to the misspelled one, "***diagnoes",*** the tri-gram will identify them as two similar text phases. As a result, the problem, ***RMA 27-93-2127 AVAIL REPEAT F/CTL ELAC 1 FAULT ELAC 1 INPUT CAPT ROLL CTL SSTU 4CE1,*** is linked to symptoms: ***WRN321, FLR1188, WRN320, WRN340,*** after matching the description to the *FB*.

3.3 Identifying the solutions

Gavin a snag message, $snag_i \subseteq SB$, we also need to determine the solution ***(S)***. In other words, the task of the solution identification is to extract repair action and equipment information from the snag message using NLP techniques [3] [14]. In general, the free text of the repair action description in the snag message contains one or more "sentences" with extensive use of acronyms and abbreviations, omission of certain types of words (such as the definite article), and numerous misspellings and typographic errors. Extracting the required specific information, namely the pieces of equipment involved in the repair and the actions performed on the equipment (replace, reset, repair, etc.), from the free text is a typical natural language understanding procedure as shown as Figure 1.

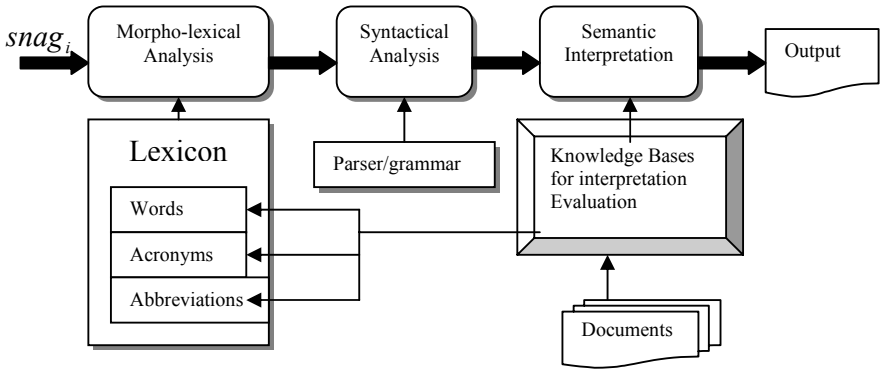

Figure 1. Main function diagram of NLP

In the natural language understanding procedure, the unstructured free text that describes the repair action is first preprocessed to determine the nature and properties of each word and token against the lexicon which contains the words, the acronyms and the abbreviations. Then the sequence of morphologically analyzed items is syntactically analyzed with a parser and checked against a grammar that describes the patterns of valid propositions. Finally the result of the syntactic parsing is semantically interpreted to generate the class of repair action and the equipment on which the action is performed. For example, the free-text that describes the repair action in the snag message, **" *#1 EIU replaced***", is analyzed as shown as Table 3.

Table 3. The result of NLP for snag example

Attribute Name of Solution *(S)*	Value
Part name	EIU
Part number	3957900612
Repair action	REPLACE
Part series number	3-25-8-2-40D

3.4 Creating a potential case

Having ***(p)*** and ***(S)*** obtained from the previous steps**,** this process creates a temporary case, $C_{tmp} = ((p),(s),(m))$. We have to check this potential case to determine if the symptoms related to the problem have disappeared or not during a period of time (window size) after the repair actions were taken. The window size is set by aircraft fleet maintenance requirements. We assume that if the symptoms of the problem disappear for the specified period (window size) that the repair was successful and the case is labeled as a positive case, otherwise it is labeled as a negative one. For example, a potential case shown as Table 4 is created from Table 2 by identifying the symptoms and solutions for the problem.

Table 4: A potential case created from Table 2 and FEO database

Case ID	*Case-1*
Case creation date	*2002-04-05*
Event date time	*1998-01-22 14:07:00*
Snag number	*M1003286*
Case quality	*Success*
Success times	*1*
Failure times	*0*
Symptoms	*WRN321 FLR1188 WRN320 WRN340*
Problem description	*RMA 27-93-2127 AVAIL REPEAT F/CTL ELAC 1 FAULT ELAC 1 INPUT CAPT ROLL CTL SSTU 4CE1*
Fin number	*222*
Repair station	*YWG*
Repair date	*1998-01-30 16:00:00*
Repair actions	*Remove/Install (replace)*
Equipment (No)	*27-92-41-501*

3.5 Maintaining the case base

The case base maintenance process implements the basic functions for case base management to determine the attributes of (m). The first set of functionality includes detecting any redundancy or inconsistency for the potential case against the existing case base. In effect we determine whether this case is similar to cases within the existing case base or not. The second set of functionality involves adding a new case to the case base, updating an existing case in the case base, deleting a case and merging multiple cases into a new case. If a potential case is new, it will be added to the case base and the case base management information

will be refreshed. If it is similar to an existing case, we have to modify the existing case by updating the case management information *(m)* or merge them into a new case. For example, if we detected a similar case (c_i) in the existing case base against the potential case c_{tmp}, i.e. $(p)_i \cong (p)_{tmp}$ [2] and $(s)_i \cong (s)_{tmp}$, then $(m)_i$ will is updated to reflect the effect of the repair action applied to the problem. If c_{tmp} is a positive case, then we increase the count of successful repair actions of $(m)_i$ otherwise we increase the count of unsuccessful repair actions of $(m)_i$.

4. Implementation

The proposed framework has been applied to the IDS project for authoring the cases from the aircraft fleet maintenance historic data (snag database) and the FEO database. We developed a Java-based CBR engine, and an automated case creation system, which incorporates the CBR engine, natural language processing, free-text matching, and database technologies. The goal of the ACCS tool is to demonstrate that we can author an set of cases in an automated way that will enhance the decision making process of the maintenance technicians.

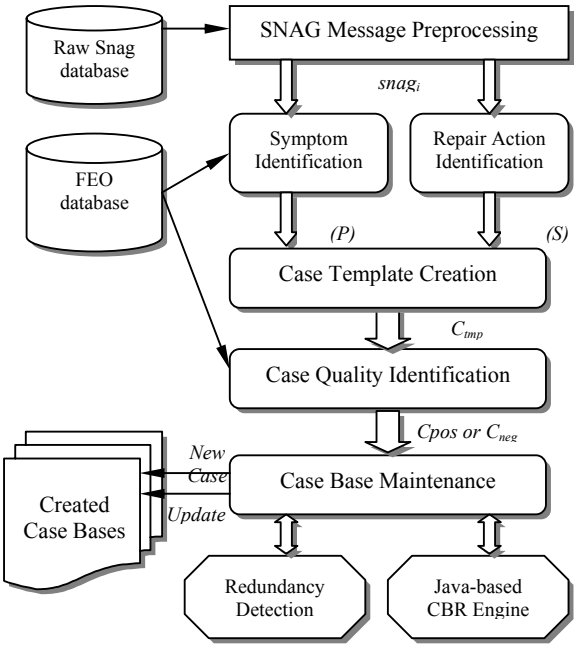

Figure 2: ACCS system implementation

[2] $(p)_i \cong (p)_{tmp}$ *means that the problem description in case* C_i *is similar to one in the potential case* c_{tmp}.

The ACCS, as shown in Figure 2, identifies the five main components: snag message preprocessing, symptom identification, repair action identification, potential case creation, and case base maintenance. The potential case creation component contains two modules: case template creation and case quality identification. The repair action identification component contains three NLP modules: the lexicon, the parser/grammar, and a knowledge base for interpretation evaluation. The component of case base maintenance is supported by the Java-based CBR engine and the redundancy and inconsistency detection modules. We have used JDK2.0, JDBC, Oracle7.0, and Prolog as development environment.

5. Experimental Results

To test the effectiveness of the proposed framework, the experiments were carried out using the developed ACCS. First we asked a domain expert to manually author the cases using the SROV. The domain expert created cases from 352 historic snag messages that were generated in IDS from Jan. 1, 1998 to Jan. 31, 1998. The cases were created in several sessions in order to reduce the influence of fatigue. The times from the different sessions were summed. Then we used ACCS to automatically author the cases from the same snag messages. Figure 3 shows the results of experiments for creating the cases manually and automatically. From the results, we found that ACCS creates almost the same cases from the same snag messages with much less time, suggesting that ACCS can create the cases quickly and correctly. It is interesting that not each clean snag message contains the completely useful information for creating a potential case because either the symptoms are not found from the FEO database, or the fix does not exist in the snag message. In the 35 constructed cases, 21 cases are created from a single snag message and consist of a positive case or a negative case; 14 cases are linked to multiple snag messages, which recorded similar resolutions for similar problems or the same problem, and they contain information on the successful or failed repair action by the attributes of case base management (m). From the statistical results, 45 snag messages from 359 snag messages were linked to those 14 cases. In total, 66 clean snag messages among 359 snag messages were useful for creating the cases.

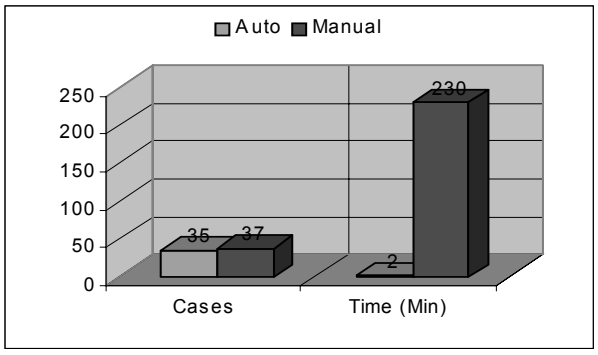

Figure 3, the experimental result comparison

6. Conclusions

In this paper, we first presented the proposed framework for automatically authoring cases from the historic maintenance data in CBR applications, and then we described its implementation, an automated case creation system for the IDS, and discussed the experiment results. From the experimental results, it can be pointed out that the proposed framework is feasible and effective for automatically authoring cases in CBR systems and it can significantly reduce the effort required. From the experimented result, we also found that it is necessary to provide an interactive environment for the domain expert to evaluate any authored cases before they are incorporated into CBR systems such as IDS. How to evaluate the cases is a very difficult task. We will work on this issue in our future work.

Acknowledgements

Many people at NRC have been involved this project. Special thanks go to the following for their support, discussion and valuable suggestions: M. Halasz, R. Wylie, and F. Dube. We are also grateful to Air Canada for providing us the aircraft fleet maintenance data.

References

1. Lehane, M., Dubé, F., Halasz, M., Orchard, R., Wylie, R. and Zaluski, M. (1998) *Integrated Diagnositic system (IDS) for Aircraft Fleet maintenance,* In Proceedings of the AAAI'98 Workshop: Case-Bases Reasoning Integrations, Madison, WI.
2. Wylie, R., Orchard, R., Halasz, M. and Dubé, F. (1997) *IDS: Improving Aircraft fleet Maintenance*, In Proceeding of the 14th National Conference on Artificial Intelligence, Calif, USA, pp.1078-1085
3. Farley, B. (1999) *From free-text repair action messages to automated case generation*, Proceedings of AAAI 1999 Spring Symposium: AI in Equipment Maintenance Service & Support, Technical Reprot SS-99-02, Menlo Park, CA, AAAI Press, pp.109-118
4. Portinale, L. and Torasso, P. (2000) *Automated Case Base Management in a Multi-model Reasoning System,* In Proceedings of Advances in case-based Reasoning: 5th European Workshop, EWCBR 2000, Trento, Italy, pp.234-246
5. Minor, M. and Hanft, A. (2000) *The Life Cycle of Test cases in a CBR System,* In Proceedings of Advances in case-based Reasoning: 5th European Workshop, EWCBR 2000, Trento, Italy, pp.455-466
6. Ferrario, M. A. and Smyth, (2000) *Collaborative Maintenance—A Distributed, Interactive Case-based Maintenance Strategy,* In Proceedings of Advances in case-based Reasoning: 5th European Workshop, EWCBR 2000, Trento, Italy, pp.393—405
7. Shiu, S.C.K. , Sun, C.H., Wang, X.Z. and Yeung, D. S.(2000) *Maintaining Case-Based Reasoning Systems Using Fuzzy Decision Trees,* In Proceedings of Advances in case-based Reasoning: 5th European Workshop, EWCBR 2000, Trento, Italy, pp.258—296
8. Smyth, B. (1998) *Case-Based Maintenance*, In Proceedings of the 11th Intl. Conference on Industry and Engineering Applications of AI and Expert Systems, Castellon, Spain
9. Smyth, B. (1995) *Remembering to Forget: A Competence Persevering Deletion Policy for Case-Based Reasoning Systems*, In Proceedings of the 14th Intl. Joint Conference on AI, Morgan-Kaufmann, pp.377-382

10. Zhu, J. and Yang, Q. (1999) *Remembering to Add: Competence Persevering Case-Addition Policy for Case-Base Maintenance*, In Proceedings of the 16th Intl. Joint Conference on AI, Stockholm, Sweden, pp.234-239
11. Racine, K. and Yang, Q. (1996) *On the Consistency Management for Large Case Bases: The Case for Validation*, In Proceedings of AAAI-96 Workshop on Knowledge Base Validation
12. Aha, D.W. and Breslow, L.A. (1997) *Refining Conversational Case Libraies*, In Proceedings of Int'l Conference of Case-based Reasoning, RI, USA, pp.267-278
13. Portinale, L., Torasso, P. and Tavano, P. (1999) *Speed-up, Quality and Competence in Multi-modal Case-based Reasoning,* In Proceedings of 3rd ICCBR, LNAI 1650, Springer Verlag, pp. 303-317
14. Ferlay, B. (2001), Extracting information from free-text aircraft repair notes, Artificial Intelligence for Engineering Design, Analysis and Manufacture, Cambridge University Press 0890-0604/01, p.p.295-305