A Proposal to Combine Probabilistic Reasoning with Case-Based Retrieval for Software Troubleshooting

Akihiro Shinmori

INTEC Systems Laboratory Inc.
3-23 Shimoshin-machi, ISL Bldg.
Toyama, 930-0804 JAPAN
shinmori@isl.intec.co.jp

Abstract

Analysis of a real case of software troubleshooting made it clear that probabilistic models, or Bayesian networks are suitable for modeling software troubleshooting processes in today's computing environments. As a way to support software troubleshooting by reusing skillful support engineers' expertise, I propose to use Bayesian networks and probabilistic reasoning for case-based retrieval. The outline of query reformulation using Bayesian network is presented.

Keywords: Bayesian network, case-based retrieval, software troubleshooting

Introduction

A lot of software troubles are happening in today's multi-vendor computing environments. While current computing environments have much functionality and flexibility, they are too complex and are vulnerable to software troubles. The rapid speed of change such as new product introductions and frequent version-ups makes the situation worse.

Engineers working on software troubleshooting have to deal with various kinds of uncertainty. Problem descriptions conveyed from customers or other support engineers are often partial and incomplete. Information they can use and rely on is scattered over multiple sources and in multiple forms such as product manuals and support information documents provided by the vendors, internal support documents compiled in-house, and various Web pages.

While there is no single almighty expert in customer support and troubleshooting, it is observed that skillful engineers with appropriate knowledge and expertise in the domain are far better at solving the problems. They can analyze the situation, ask appropriate questions to clarify the situation, search the case-base effectively, and come up with a solution in a reasonable time frame. If it would be possible to formalize those engineers' knowledge in reusable forms, it would be possible for less skillful engineers or other engineers with different skills to take advantage of them.

Our research group has been working to develop a support system for software troubleshooting. This paper reports an analysis of a real case of software troubleshooting process and introduces a proposal to combine probabilistic reasoning with case-based retrieval.

Analysis of Software Troubleshooting – A Case Study

We have analyzed a real case of software trouble at our parent company (referred as "the company" in the followings) (Shinmori and Shibagaki 1997).

The whole story

The company developed and sold an application called X to several customers. X runs on a Microsoft Windows client and creates a data file on a Novell NetWare server. The problem was that the file X creates on the NetWare server got partially corrupted at very rare intervals on a customer's site.

The customer called the company and told the problem to the front-line engineer. The front-line engineer did not find the cause or similar past cases on the spot and forwarded this case to a back-line engineer (referred as "the troubleshooting engineer" in the following). The troubleshooting engineer at first thought the culprit was the X itself. But the efforts to reproduce the problem at the testing environment did not succeed. The engineer asked the customer to provide more detailed information.

After several sessions, the customer reported that the problem tends to happen when there is little free disk space on the NetWare server, specifically when most of the disk space on the server is occupied by the "space available from deleted files," [1] and the network traffic is high. After testing further and inspecting source programs, the engineer found that one portion of the programs lacked write error checking. It could be a valid cause of the problem, but it alone did not explain why the problem happened at very rare intervals and only at one customer's site.

The engineer continued investigation assuming that there were other causes. He doubted the NetWare client and the NetWare server. He searched the case-base provided by the

[1] The area managed by NetWare OS to recover deleted files.

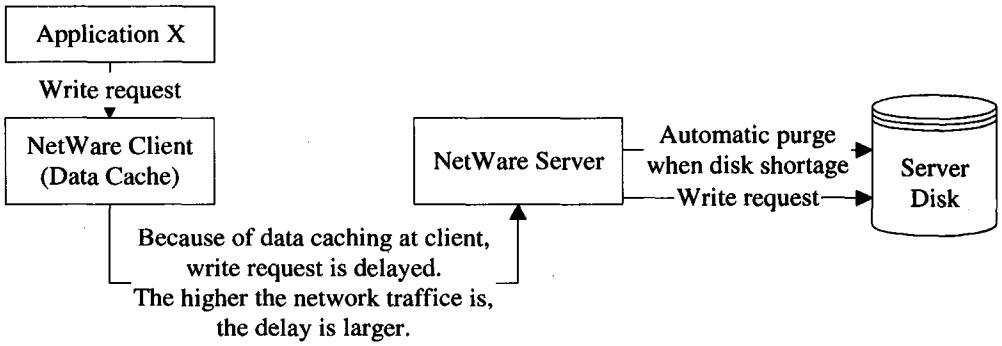

Figure 1. Background Mechanism of the Problem

vendor ("Technical Information Document" provided by Novell Inc.). But searching by using the keywords of "server", "file", "write", "error", and their combinations always generated too much hits.

While looking through the hit list and browsing the candidates, he by chance found a case that was describing, "In a certain setting of NetWare client, running an application Q may cause data loss."

While reanalyzing the situation, he refocused on the additional information from the customer. Then he searched the manual and found the following limitation of NetWare client:

> **CACHE WRITES**
>
> Sets the cache write requests to ON or OFF.
>
> Setting this parameter to OFF increases data integrity but decreases performance. Leaving this parameter set to ON can cause data loss if the NetWare server runs out of disk space between write requests.
>
> Default : ON
>
> (quoted from "NetWare Workstation DOS and Windows" in "NetWare 3.12 Manual Set" by Novell Inc.)

He also found that the default value of "CACHE WRITES" was off in the previous version of NetWare.

Considering the fact that X was developed before the previous version of NetWare, the assumption described in Figure 1 explained the whole story well. Several months had passed since the first call was made.

The engineer advised both the customer and the front-line engineers to set the "CACHE WRITE" option off as a workaround. He also notified the developers to fix the source code.

Modeling by Bayesian Network

The above troubleshooting process can be modeled by using probabilistic model, or Bayesian network. Bayesian network is a theoretical framework based on probability theory (Pearl 1997) (Charniak 1991) (Jensen 1996). It is suitable for dealing with uncertainty and several probabilistic inferences can be done on it. It has been used in a variety of applications, most successfully in diagnostic applications.

Bayesian network consists of nodes and directed arcs. With nodes, variables are attached usually representing discrete states such as true or false, relevant or irrelevant. Nodes without parents are called root nodes and assigned prior probability distributions with their variables. With non-root nodes, conditional probability distributions are specified given their parents' values combinations.

In diagnostic applications, directed arcs connecting nodes usually represent causal relationships from causes to

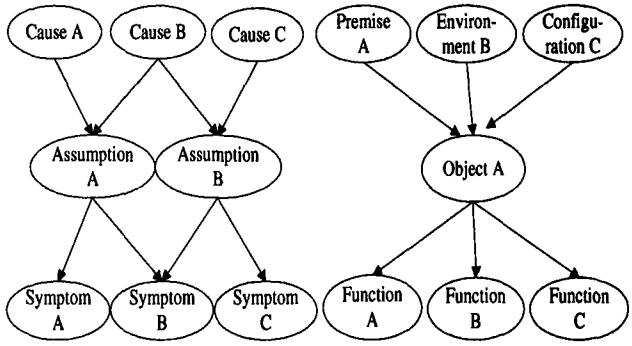

Figure 2. Bayesian Network for Software Troubleshooting

symptoms. While the notion is also applicable to the domain of customer support and software troubleshooting, additional relationship is needed to represent that a certain object shows some functionality in certain conditions. For example, a certain version of a certain DLL (dynamic link library) behaves in a certain way with certain other DLL's.

Because most of the objects in today's computing environments are "COTS" (Commercially Off-The-Shelf), these kinds of knowledge or know-how are inherently uncertain and informal. Nevertheless, they are playing an important role in customer support and software troubleshooting. They are typically accumulated through the engineer's own experience or word-of-mouse communication from others.

To model uncertain knowledge related to software troubleshooting, I propose to use Bayesian network as shown in Figure 2. The right half shows the "object-functionality relationship" I discussed in the above.

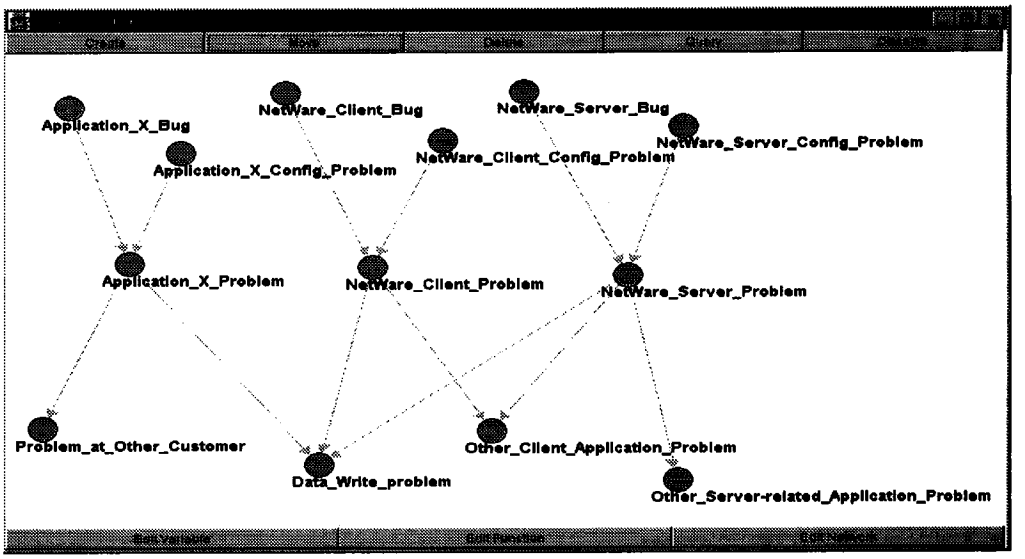

Figure 3. Bayesian Network modeling Troubleshooting Process (drawn in JavaBayes v0.333)

Analysis of the troubleshooting process

The previous troubleshooting process can be modeled by Bayesian network. Figure 3 is a screen image of the Bayesian network model drawn by using the JavaBayes package (Cozman 1997).

To reduce the complexities of the model and to ease the knowledge acquisition task from experts, the model is constructed following the notion of "noisy-or" model (Pearl 1997) (Jensen 1996). The probability distribution is set follows:

- For each root node, the prior probability to take true value is set 0.
- For each non-root node, conditional probability to take true value is set as follows where the number of parent nodes which take true value is n:

$$1-(0.2)^n \quad (n \geq 1)$$
$$0.1 \quad (n = 0)$$

The process can be modeled as follows:

1. As the first call indicated file write error, observe the "Data_Write_Problem" node as true.
2. At this point, posterior probabilities for the three middle nodes to take the value of true are all the same (0.28).
3. After finding that X lacked some error checking, set the prior probability of the "Application_X_Bug" higher (true with 0.80). Then, among the three middle nodes, the "Application_X_Problem" node gets the highest posterior probability (true with 0.87). The remaining two get 0.15.
4. Because no other customers report the problem, observe the "Problem_at_Other_Customer" as false. Then, the posterior of "Application_X_Problem" node gets lower (true with 0.60) and the remaining two get 0.21.
5. After finding a relevant case in the case-base, set the prior probability of true value of "NetWare_Client_Config_Problem" higher (0.80). Then, the node "Application_X_Problem" gets lower value (true with 0.40), the node of "NetWare_Client_Problem" gets higher value (true with 0.82), and "NetWare_Server_Problem" gets slightly lower (true with 0.13).

The model described in Figure 3 and used in the above analysis is not just a hindsight record of the process but can be viewed as the know-how of the troubleshooting engineer. By creating and saving the models, they can be reused for another troubleshooting sessions or by less experienced engineers.

Combining Probabilistic Reasoning with Case-Based Retrieval

Several contributions have already been made to improve case-based retrieval mainly for help desk applications (Simoudis 1992) (Shimazu, Shibata, and Nihei 1994). But the uncertainty aspect of troubleshooting analyzed in the previous section has not been addressed well in them.

Bayesian network is already used in software troubleshooting (Breese and Heckerman 1995). It is also used in hardware troubleshooting in copier machines (Hart and Graham 1994). These approaches assume that complete Bayesian networks are constructed beforehand. The target problem domain is rather narrow and restricted.

Query Reformulation using Bayesian Networks

The most obvious way to use probabilistic reasoning in software troubleshooting is to create full and complete

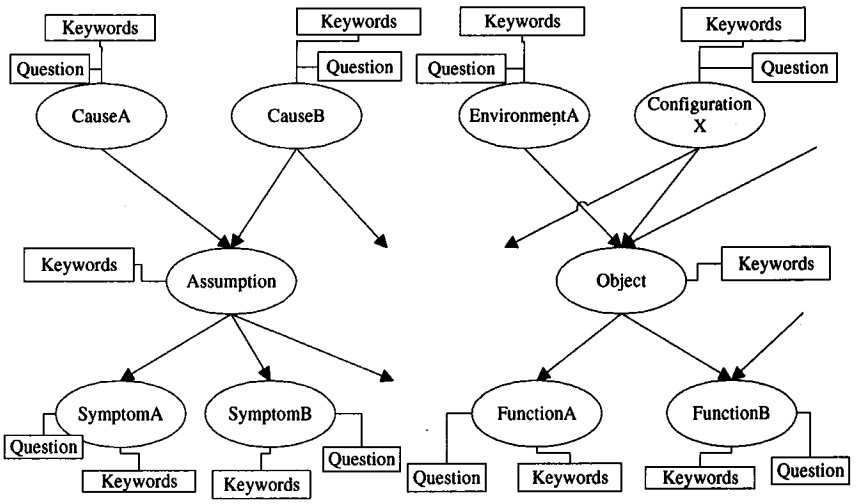

Figure 4. Bayesian Network for Query Reformulation

Bayesian networks and to diagnose the problems based on them. However, it is not a trivial task because it requires complete analysis and deep understanding of the domain.

Instead, I propose to use Bayesian networks to reformulate query for case-based retrieval as follows.

Model Construction:

- Create (partial) Bayesian networks which model domain knowledge by incorporating experienced engineers' knowledge or by extracting from existing case-base.
- Each node is assigned keywords that represent it.
- Questions are attached to bottom nodes that are used to observe the node.
- Questions or some other means are attached to the root nodes that are used to observe the node or change the prior probabilities.

The conceptual model is shown in Figure 4.

Query Reformulation Outline:

- Load the appropriate Bayesian networks.
- Analyze the user's initial problem description and search for the nodes in the networks which have some kind of similarities (such as linguistic similarities).
- Provide the user with questions attached to the nodes. Observe the nodes appropriately based on the answers given by the user.
- Calculate posterior probabilities of the middle nodes.
- Retrieve the keywords of the observed nodes and the middle nodes that have larger values than a certain threshold. Generate a query by collecting the keywords.
- If the target case-base system supports weighted query term, use the posterior probabilities as the weight of the keywords.
- Search the case-base. The user can iterate the above process until he or she finds the appropriate case.

Implementation of Prototype System

As a long-term research goal, we plan to develop a "call-avoidance system" which the customers themselves use to solve problems (Andrews 1996). So, we are developing a Web-based prototype system.

For the purpose of processing Bayesian network and reasoning probabilistically, we used the engine derived from the JavaBayes package. From v0.333, JavaBayes supports XML (W3C 1998) and the Bayesian networks created by the graphical editor can be saved in XML files.

At this point, we have implemented an initial version of a query reformulating Java applet. It analyzes initial problem description, load Bayesian network stored in a XML file, asks questions, does Bayesian inference, and reformulates the query. It can be used to call general-purpose search engines on the Internet as well as our special-purpose database search servlet. The conceptual diagram of the system is shown in Figure 5. In Figure 5, the shaded components are the systems we are developing.

Sample Scenario Using the System

In the following, a sample scenario using the system is explained.

It is well reported that various software troubles happened after installing the latest version of Internet Explorer (IE4.0) on Windows95 OS[2]. Specifically, it is reported that the dialog of cover page setting in "Microsoft FAX" program got corrupted. In fact, this trouble happened because among the several DLL files installed with IE4.0, different versions of "awfxcg32.dll" can interfere with oth-

[2] It is reported much especially in the Japanese environment.

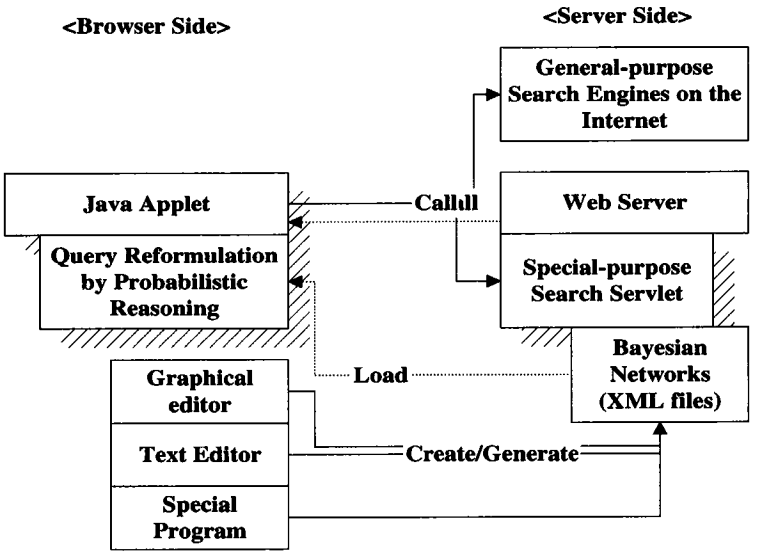

Figure 5. Conceptual Diagram of the System

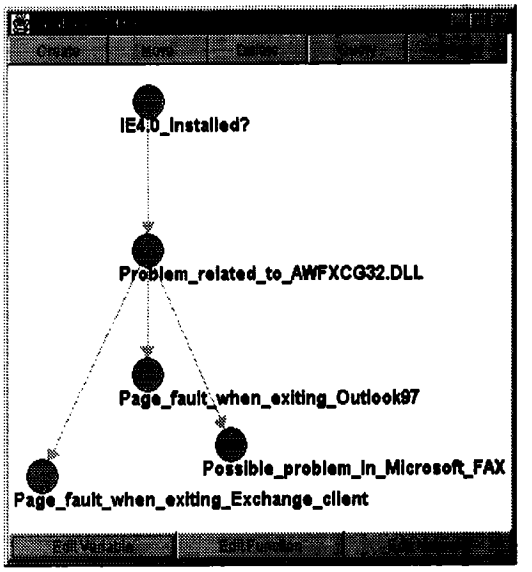

Figure 6. Sample Bayesian Network

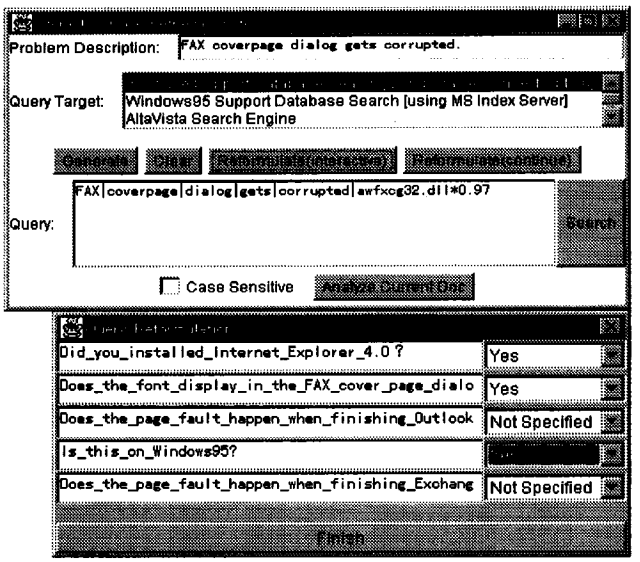

Figure 7. Sample Screen Image of the Applet

er programs such as Microsoft FAX.

When considering the troubleshooting process by a novice engineer, the followings can be assumed:

1. In the case-base, there is no exact case describing the problem situation.
2. The novice engineer does not know that the direct cause is in the awfxcg32dll. Nor does he/she know that the root cause is in the previous action: installation of IE4.0.
3. The experienced engineer already knows that depending on a different version of awfxcg32.dll other problems, such as (Microsoft 1998), can happen.
4. The experienced engineer also knows that awfxcg32.dll is used by Microsoft FAX.

In this case, the novice engineer tends to get stuck in the early stage of troubleshooting because he/she cannot find relevant cases by just using the keywords he/she describes the situation. If the experienced engineer's knowledge is encoded in a Bayesian network as in Figure 6 and is used by the query reformulating Java applet, the novice engineer can generate a query to retrieve relevant cases. Figure 7 is a screen image of the Java applet.

Discussion and Issues to Investigate

The proposed approach is more open-ended than those described in (Breese and Heckerman 1995) and (Hart and Graham 1994) because Bayesian network is just used to reformulate query to search various case-bases. Although it lacks diagnostic powers, it is flexible.

Query reformulation mechanism in the general context of information retrieval is reported in (Gauch and Smith 1991). They used rule-base technology and thesaurus to expand or narrow the query. Causal knowledge typical in software troubleshooting cannot be incorporated into it.

As the next step of research, we will refine and evaluate the prototype system in more realistic contexts. We need to investigate the following issues.

- How do we create and maintain Bayesian networks?
- How do we share and exchange them?

Acknowledgments

The author would like to thank Dr. Kouichiro Ochimizu and Dr. Youichi Shinoda of Japan Advanced Institute of Science and Technology for useful discussions. The author

also thanks Mr. Fabio Cozman of Carnegie Mellon University for developing and providing the JavaBayes package.

References

Andrews, D. 1996. Self-Help Software to the Rescue. *BYTE.* October. 1996: 26-27.

Breese, J. S. and Heckerman, D. 1995. Decision-Theoretic Case-Based Reasoning. Technical Report. MSR-TR-95-03. Microsoft Research.

Charniak, E. 1991. Bayesian Networks without tears. *AI Magazine.* Winter. AAAI: 50-63.

Cozman, F. 1997. JavaBayes v0.33. http://www.cs.cmu.edu/~javabayes.

Gauch, S. and Smith, J. B. 1991. Search Improvement via Automatic Query Reformulation. *ACM Transactions on Information Systems.* 9(3):249-280.

Hart, P. E. and Graham, J. 1994. Query-free Information Retrieval. In Proceedings of the International Conference on Cooperative Information Systems.

Jensen, F. V. 1996. *An Introduction to Bayesian Networks.* New York, NY: Springer-Verlag.

Microsoft. 1998. "Error Message: Exchng32 Caused an Invalid Page Fault..." Microsoft Support Online. http://support.microsoft.com/support/kb/articles/q174/8/87.asp

Pearl, J. 1997. *Probabilistic Reasoning in Intelligent Systems.* San Francisco, CA: Morgan Kaufmann. (Originally published in 1988).

Shimazu, H., Shibata, A., and Nihei, K. 1994. Case-Based Retrieval Interface Adapted to Customer-Initiated Dialogues in Help Desk Operations, Proceedings of AAAI'94

Shinmori, A. and Shibagaki, Y. 1997. Support Process Improvement Approach based on the Analyses of Software Troubleshooting Processes. Proceedings of Software Symposium'97. (in Japanese)

Simoudis, E. 1992. Using Case-Based Retrieval for Customer Technical Support. IEEE Expert. :7-12.

W3C. 1998. Extensible Markup Language (XML) 1.0. World Wide Web Consortium. February 1998. http://www.w3.org/TR/1998/REC-xml-19980210

Appendix: Questions and Answers Concerning CBR Integrations

1. Integration name/category:

- Interactive case retrieval system incorporating query reformulating based on Bayesian network

2. Performance Task:

- Probabilistic reasoning and case-based retrieval in customer support and troubleshooting for software troubles

3. Integration Objective:

- Bayesian network assists in retrieval by query reformulation

4. Reasoning Components:

- Bayesian network for query reformulation support

5. Control Architecture:

- CBR as master

6. CBR Cycle Step(s) Supported:

- Retrieval

7. Representations:

- Textual cases. Bayesian network created beforehand and stored in XML files.

8. Additional Components:

-Ask questions to the user to observe the node in Bayesian networks

-Generate query adapted to the target search system

9. Integration Status:

- Proposed and currently being implemented.

10. Priority future work:

- Application and evaluation