AN AUTOMATED KNOWLEDGE ACQUISITION SYSTEM FOR MODEL-BASED DIAGNOSTICS

Gregory B. Winter
Prospective Computer Analysts, Inc.
1800 Northern Blvd.
Roslyn, New York 11576

ABSTRACT

This paper discusses how knowledge acquisition will become a key element to improving the design, production, and testing of complex electronic, mechanical, and hydraulic systems. Specifically, knowledge acquisition will become critical to the use of dependency model systems such as the Weapons System Testability Analyzer (WSTA), the Test Generator for Inferred Reasoning (TGIR), and other such systems. The paper also discusses a means for automating knowledge acquisition and integrating it into the design process. It then presents NASA's CAD/CAE Knowledge Base Development Tool (KBDT) as a prototype for demonstrating the concept of automated knowledge acquisition. KBDT's basic structure, operation, and its integration with NASA's Knowledge-based Autonomous Test Engineer (KATE), are described. The paper also discusses KBDT's unique combination of voice synthesis, voice recognition, natural language processing, and knowledge acquisition processing components, integrated via a blackboard architecture, necessary for this knowledge engineering application.

INTRODUCTION

The increasing complexity of future electronics systems will soon exceed their ability to be tested and maintained unless new tools are found to improve design methods, testability, diagnostics, and test programs. Fortunately, diagnostics can be automated by specialized expert systems, called dependency or causal model-based systems which infer failures as causes for observed symptoms. The most promising of these reason over a symbolic, qualitative model of a physical device and infer failure conditions from the device's structure, behavior, and causal relationships. The inferencing can be extended to generate test strategies using a next-best-test algorithm using weighted dependencies. Examples of such model-based systems include WSTA (as part of the Integrated Diagnostic Support System or IDSS), LOGMOD, STAMP, TGIR, I-CAT, and KATE.

The drawback to these systems is that the dependency models must be designed or augmented by knowledge engineers, not by the original electronic, hydraulic, or mechanical engineers who designed the system. To create a complete model, knowledge engineers must interview the designers, analyze the schematics, and ask detailed questions to develop the model of the system. This process is tedious and time-consuming, requires interviews, taped working sessions, and extensive analyses, and is often prone to error and misinterpretation. Although much of the dependency model can be generated automatically from Computer-Aided Design and Engineering (CAD/CAE) systems (as is done with WSTA), they do not completely capture the designer's personal knowledge and rationale needed to create an effective dependency model.

Capturing knowledge for diagnostics and maintenance testing can be automated, however. NASA's Kennedy Space Center developed the Knowledge Base Development Tool (KBDT) which automates the process using a protocol and interview technique. As the designer works on the CAD station, KBDT asks questions concerning components, connections, behaviors, and reasons for selecting those components. User responses are captured in the knowledge representations needed to build the dependency model. The system also captures design rationale and other types of knowledge that form a complete historical database of knowledge about the system.

The Impact of Automated Knowledge Acquisition

The impact of automated knowledge acquisition will be immeasurable. In addition to schematics, technical manuals, and parts lists generated for a system, the knowledge and experience that went into designing the system can be automatically captured and stored. This knowledge can then be used for two purposes. First, it can be used to create dependency models for use by WSTA, TGIR, I-CAT, STAMP, and other tools for analyzing

CH3148-4/92/0000-0167 $1.00©1992 IEEE

testability and creating diagnostic test strategies. Second, the knowledge can be used as a repository for detailed design rationale. This knowledge can then be retrieved at any time to help manufacture, redesign, test, or maintain the system.

Design rationale can be examined by test engineers to determine why the designer selected certain parts, why certain structures were created, and how components are expected to interoperate. For example, a circuit may be found to have an extra multiplexer with no relation to the function of the circuit. Only through direct dialogue with the designer does it become known that the multiplexer was included for the sole purpose of increasing the number of test points in the circuit.

With this type of knowledge, new designers, test engineers, and production engineers who know little about the system will be able to inquire, ask questions, and, in a sense, interview the best expert for the system, the original design engineer.

However, knowledge acquisition must be automated if it is to achieve these goals. Manual knowledge acquisition is too time consuming for it to become an integral part of the design process. Too often it becomes an after-the-fact task which the designer is unwilling to perform. If performed by knowledge engineers during the actual design process, the designer will resent it as an interruption and an obstacle to completing the design.

Thus, automation is the key to successful knowledge acquisition and, by extension, the development of effective diagnostics using dependency model systems. An automated system that acquires knowledge directly from CAD/CAE files and from the designer will become another tool in the vast array of tools in the engineering environment of the future.

The remainder of this paper describes a seminal effort to create an automated knowledge acquisition system. This system, the CAD/CAE Knowledge Base Development Tool (KBDT), was designed to capture design rationale from both Computer-Aided Design and Engineering (CAD/CAE) database files and the design engineer. It is designed to capture information and knowledge about an evolving design that is above and beyond the data represented in conventional documentation or CAD/CAE database files.

KNOWLEDGE BASE DEVELOPMENT TOOL

KBDT was developed with funding from the Small Business Innovative Research (SBIR) program. KBDT's initial design was first specified as the result of a one year Phase I SBIR effort begun in 1987. Actual development began in July 1989 under a two year, Phase II effort. The KBDT design met all of its technical objectives and was demonstrated on April 17th, 1991. Both SBIR efforts were administered and evaluated by NASA at Kennedy Space Center under the direction of the Artificial Intelligence group in the Advanced Launch System (ALS) Program. The system was also developed in cooperation with the Jet Propulsion Laboratory in Pasadena, California.

The real world domain selected for KATE and KBDT is the design and test of the Liquid Nitrogen (LN2) tanking system for the Advanced Launch System (ALS). This domain was sufficiently well defined to constrain the vocabulary and problem space enough to create a realistic knowledge acquisition system capable of developing full-fledged dependency models.

General Functional Operation

KBDT works interactively with the design engineer during the design process. KBDT captures knowledge about design rationale using information supplied by both the designer and the CAD/CAE workstation. Thus, inputs to KBDT include both the CAD/CAE files themselves and verbal responses from the designer.

KBDT currently uses CAD files created and residing on an Intergraph CAD Microstation. From these files, KBDT's Model Builder program extracts a structural model of the system and outputs it to the Knowledge-based Autonomous Test Engineer (KATE). KATE is a dependency model-based diagnostic tool for autonomous control and monitoring for automated process control which was developed by NASA concurrently with KBDT. KATE was programmed in LISP and is hosted on a Symbolics Computer.

As KBDT's Model Builder creates the causal model, it identifies areas requiring additional knowledge, either about the structural design or the rationale for creating the design. Changes made to the design are also identified, and the design rationale for making those changes is

sought. In this way, KBDT creates a set of questions to solicit information from the designer about the design. The questions created by KBDT are first structured by a natural language subsystem to create naturally sounding sentences. These questions are then voiced to the designer via a voice synthesis system.

The designer responds verbally to the questions posed by KBDT. Responses are recorded, parsed, and information extracted into data structures which are then analyzed for their knowledge content. Altogether, this subsystem of KBDT which voices questions and records responses is known as the Design Knowledge Capture (DKC) Subsystem.

With knowledge captured in specific data structures, KBDT then reformats the data into knowledge representation structures which can be stored as part of the KATE model or as associated knowledge in a specific knowledge base. Some design knowledge is not necessarily dependency model specific (i.e., is unrelated to the system's structural design or functional behavior). This non model-oriented knowledge is stored in the representations most appropriate for it. KBDT provides eight such knowledge representations:

1. General Domain Knowledge
2. Specific Domain Knowledge
3. Plans
4. Heuristics
5. Analogies
6. Experiences
7. Guidelines
8. Facts

These knowledge representations have been found to capture every type of knowledge related to design rationale as described by Roger Schank [5]. More importantly, they allow different types of knowledge to be represented in their most natural form. For example, knowledge about plans should not be coerced into a heuristic representation since heuristics fail to capture the temporal requirements of plans.

Overall System Design

Figure 1 provides the overall design and functional breakdown of KBDT. KBDT utilizes the following commercially available hardware and software:

1. IBM Compatible 386 Personal Computer
2. SUN 4/260 Sparc Workstation
3. VOTAN Voice Recognition Card
4. Prose Voice Synthesis Card
5. Interactive UNIX v. 4.01 for the PC 386
6. SUN OS (UNIX v. 4.0 compatible)
7. NL Builder by Synchronetics

As shown in Figure 1, KBDT consists of several independent program modules, programmed in C and C++, residing on both a PC 386 and a SUN 4/260 workstation. All portions residing on the PC 386 make up the natural language processing portion of KBDT. The portion of KBDT residing on the SUN 4/260 consists of three main modules: the Model Builder, Design Knowledge Capture (DKC) Subsystem, and the Knowledge Base System. Each knowledge base also has an inference engine associated with it to retrieve and inference on stored information.

DETAILED OPERATION

CAD, KATE, and Dependency Model Files

CAD files from the Intergraph Microstation are input into the Model Builder Portion of KBDT which scans, parses, and reformats them into LISP knowledge structures (frames with defined class inheritances and hierarchies) suitable for use by KATE. KATE is not a formal part of KBDT and resides on a Symbolics Computer. The knowledge structures containing the model information are stored by KBDT's Model Builder and are later input into KATE. The knowledge extracted from the CAD files is primarily topological and is used to create a first order dependency model.

Model Builder System

During the creation of the dependency (causal) model, conflicts may occur where component parts do not match, components are improperly connected or insufficiently defined, etc. In these instances, information about these conflicts is sent from the Model Builder to the DKC subsystem to generate questions and solicit knowledge to resolve the conflicts. As indicated in Figure 2, an entire data structure is passed from the Model Builder program to the DKC subsystem. This data structure is analyzed to identify question templates best suited to elicit the required information.

Once the model and dependency knowledge has been solicited from the designer, it is passed back into the Model Builder using essentially the same knowledge structure originally passed to the DKC subsystem. In some cases, the DKC Subsystem

may request information from the Model Builder. Actually, this request assumes that the model has already been built. In any case, the request is intended to obtain information usually used to generate a question. For example, the system may pose the question, "Why is component A connected to component B?" This is a generic question template. In actuality, values are passed in using model information so realistic questions such as "Why is the magnetic choke connected to the output of the 250 volt power supply?" can be generated. Since this action assumes the model has already been built, the Model Builder searches through the model and retrieves the requested information. Information unavailable in the model may generate new questions that must be answered before the original request for information can be satisfied.

Design Knowledge Capture (DKC) Module

The DKC module uses a set of template questions stored in a question template library. Currently, the library contains a sufficient number of question templates (approximately 65) to conduct a working session with a NASA LN2 tanking system hydraulics engineer. Question templates are programmed as frames containing slots for names of components, actions, functions, and other information that may be filled with actual values from the model.

An interactive session is initiated using the model created by the Model Builder. Conflicts, missing information, or changes to the model are identified which trigger questions to be retrieved from the question template library. Actual values are passed from the Model Builder to the DKC and then entered into the question template. The process of selecting a question template and passing values is provided by the Acquiring Knowledge module. This module acts as the central clearing house for all questions and knowledge provided by or retrieved by the DKC subsystem. When a question is created by the DKC, the Acquiring Knowledge module is responsible for passing the data from the model into the question. It then outputs the question to the Issue Questions/Commands module which acts as a buffer to the NL Processor. The Acquiring Knowledge module also receives completed response data from the user via the Receive Responses module. The Receive Responses module acts like a buffer in that it receives data structures created by the Natural Language Processor on the PC 386 and stores them on a queue until requested by the Acquiring Knowledge module.

All questions and response data structures passed between the DKC module on the SUN 4/260 and the Natural Language Processor on the PC 386 are sent via a bidirectional serial link using the Kermit communications protocol.

The Acquiring Knowledge module retrieves response data from the Receives Responses module and reformats it into the appropriate knowledge representation. The module selects one of eight knowledge representations according to the type of knowledge it expects to receive. The module expects to receive a certain type of knowledge since it has already sent a question. Thus, each question template contains an expectation slot which identifies the type of knowledge representation most suitable for the responses to that question.

The use of expectation values with the question templates was derived from the work of Roger Schank [5] who defined Memory Organization Packets (MOPs) as the central structure for all knowledge types. Expectation values also help to constrain the need for extensive natural language processing since the system always expects certain types of response values. A question about valves would expect to receive a response somewhat related to valves.

Once the response data is reformatted into a knowledge representation, the Acquiring Knowledge module inserts the knowledge into the appropriate knowledge base. The system has been designed to accommodate a knowledge base for each knowledge representation. Although this may lead to independent islands of knowledge with conflicting rule sets, the Acquiring Knowledge module is responsible for resolving knowledge conflicts by generating additional questions. The design of the Acquiring Knowledge module was influenced by the blackboard architecture described by Buttner, et.al. [4]. This object-oriented architecture allows for multilevel reasoning across different expert systems and the seven different knowledge representations used by KBDT.

Knowledge Bases

KBDT's most unique feature is its use of eight independent knowledge bases where each knowledge base contains a particular type of knowledge. This requirement was driven by the empirical observation that not all knowledge can be represented as heuristics, frames, or a combination of the two. Roger Schank [5] has identified eight types of knowledge which are best suited for use according to their own representation. KBDT provides a knowledge base for each such possible representation. KBDT also provides an inference engine for each knowledge base to insert and retrieve knowledge. Each engine is unique to best exploit the type of knowledge to be inferenced. Applications can make use of these inference engines to reason or retrieve knowledge for a particular application. All applications can retrieve knowledge from any of the knowledge bases through the blackboard in the Acquiring Knowledge module. Essentially, a request for information is posted on the blackboard and all knowledge bases are allowed to respond.

With this approach, KBDT ensures an independence not found in other knowledge base systems. First, it is independent of any one knowledge representation since it utilizes all known types and provides separate inference engines for each. Second, the system is not biased toward any type of application since almost any one of the eight knowledge representations can be selected to support an application. Thus, for scheduling problems an application can select the plans knowledge base for its inferencing. Similarly, the heuristics knowledge base can be used for conventional diagnostics problems using backward chaining.

Natural Language Processing (NLP) Subsystem

This system provides the primary interface to the designer. Although conventional keyboard/ monitor/ mouse interfacing is available, this system provides the most non-intrusive, natural means of eliciting design rationale knowledge from the design engineer. Data structures are sent to the NLP subsystem on the PC 386 from the Issue Questions/Commands module in the DKC subsystem on the SUN 4/260. A small module within the NLP subsystem converts the questions in the data structures to sentences which can be output to the designer by the voice synthesis card in the PC 386 and a speaker. The voice synthesis card works well enough to provide clear sentences with inflection, tonal variations, and accents if desired.

The designer then responds verbally to the questions posed by the voice synthesis card. A microphone records these responses, and the voice recognition card recognizes and organizes the received phonemes into words. This card requires a brief twenty minute training session with the design engineer so the system will be able to recognize their particular voice patterns. The voice recognition card requires programming to organize each set of phonemes into words.

The voice recognition card outputs a set of words to the Natural Language Processing module. This module scans each word output by the voice recognition card and identifies each as a noun, verb, adverb, conjunction, etc. It identifies words as parts of speech using Augmented Transition Networks (ATNs) which are essentially state transition diagrams for diagraming a sentence. In this way, all parts of the response are identified.

The NLP system does more than identify the type of word. It identifies the word as being part of an expected response. As noted previously, the data structures built by the DKC subsystem in the SUN 4/260 contain expectation values; values that should be responded to by the user. If the system poses a question concerning valve stems, the system expects a response about valve stem design. Thus, responses can be understood within the context of the question. If responses are recognized which violate expectation values, KBDT may either repeat the question, or try another question along the same line of reasoning.

Word Dictionary

The NLP subsystem contains a dictionary of words and their senses which must be programmed to identify particular words in the response. This dictionary must be built for each application since each contains its own unique terminology. The dictionary must also identify the sense of each word which can be complex since some words can be used as both noun, verb, and adverb.

Once all words are identified and correctly associated with its sense and use within the context of the question, the NLP subsystem builds a data structure to package the information. This package is then sent to the Receives Responses module on the SUN 4/260. There, the information is unpacked by the Acquiring Knowledge module which reformats the information into the correct knowledge representation.

It should be noted that KBDT is an automated, not automatic, knowledge acquisition system. For each application, some initial knowledge acquisition must be performed to identify the specific vocabulary, acronyms, and euphemisms used in the application. These parts of speech must then be programmed into KBDT's dictionary prior to any actual knowledge acquisition of the system itself.

SAMPLE SESSION

The following transcript is part of a working knowledge acquisition session held during the demonstration to NASA. Questions in bold typeface were posed by KBDT. The regular typeface indicates actual user responses which were correctly parsed and recognized by KBDT.

Question 5: What type of component are you working with?
Designer: A valve.

Question 25: What is the component reference designation of this valve?
Designer: Repeat question.

Question 25: What is the component reference designation of this valve?
Designer: MCV101

Question 15:Are you
a) replacing this component?
b) removing this component?
c) adding this component?
d) moving this component?
Designer: Adding this component.

Question 20: Do you consider this component critical to the system?
Designer: Yes.

Question 35: Is this component active or passive?
Designer: Active.

Question 55: What factors, if any, did you consider in selecting this component?
Designer: Ability to control flow, temperature rating, and pressure requirements.

Question 60: Are there any other factors which you considered which were not mentioned? Please identify these factors.
Designer: The type of control would be one factor. There are two types of control. One type is proportional control. In a proportional control system, the valve area increases with the square of the valve position. In a linear control system, the valve area linearly increases with the valve position.

Question 70: What is the required range of values or required value for the flow coefficient?
Designer: 0.05 - 0.8

KBDT was demonstrated to be able to follow three main lines of questioning including knowledge capture related to adding a component, changing a component, and removing a component. Between these three lines of questioning, KBDT is capable of asking approximately 65 generic types of questions and recognizing over 750 words.

These questions are considered sufficient to capture all knowledge related to the structural design of the LN2 tanking system's hydraulics. In addition to the capture of design rationale knowledge, KBDT can be used to capture information needed to actually build the KATE compatible model of the system. For example, by using the **adding a component** line of reasoning, the system can capture knowledge about each component, its attributes, and its connections. This knowledge is then formatted into LISP compatible dependency model files for KATE. Figure 1 below shows an example of a frame-like knowledge representation generated by KBDT from an adding a component session:

Figure 1
Sample Model Component Knowledge Representation

```
(object PUMP
 (type SOURCE)
 (inputs ((ELECTRICAL-SOURCE ANALOG-
   VALUE)
   (FLOW-INPUT ANALOG-VALUE)
   (PRESSURE-INPUT ANALOG-VALUE)))
 (outputs (FLOW-OUTPUT ANALOG-VALUE)))
 (output-functions (FLOW-OUTPUT (IF (AND
    ELECTRICAL-SOURCE FLOW-OUTPUT)
    (SQRT (/ (- MAXIMUM-PRESSURE
    PRESSURE-INPUT) (+ PUMP-CV
    TOTAL CV 0.0)
 (parameters ((PUMP-CV 0.05)(MAX-
    PRESSURE 0.8)))
 (strong-constraint (TYPE-OF-OUTPUTS TYPE))
 (normal-constraint (UNITS))
 (weak-constraint (RATING DELAY))
 (units (FLOW-INPUT GPM)))
```

The knowledge frames captured by the DKC system are translated by the Model Builder into the dependency model representations needed by KATE. The Model Builder in this instance is little more than a formatter and is easily modified to output data to other model-based reasoning systems if the knowledge base consists of component representations mapped directly to components of a system schematic database.

CONCLUSIONS

As part of its development, the KBDT prototype system was used with actual engineers at NASA's Kennedy Space Center who are designing a new type of tanking system using liquid hydrogen and oxygen. During these sessions, KBDT was found to require only twenty to thirty minutes of training to initialize the voice recognition system for a new engineer. Besides this, it requires very little additional training or time for set up and first use.

NASA engineers who have used KBDT have been enthusiastic about it and have found its combination of voice synthesis and recognition exciting to work with. More importantly, they view the system as an important new tool for imparting knowledge about their design that is important not only for NASA but themselves. In fact, we have found that automated knowledge acquisition presents new opportunities for improving the design cycle since designers can refer to previous analyses and rationale which they may have "forgotten".

Research remains to be done, however, especially in the areas of causal modeling, qualitative reasoning, and natural language processing. In fact, a number of issues must be researched before a system like KBDT can be mature and robust enough to capture knowledge in larger real-world domains. First, the template method severely limits the range and realism of the natural language processor. At some point, it will be necessary to invoke qualitative reasoning to perform some portion of the natural language process. Second, the approach used by KBDT assumes that the engineer's responses are correct or complete. This assumption is incorrect for most other real-world applications since responses will normally contain inconsistencies, incorrect causal attributions, or causal impossibilities.

Lastly, before an ideal knowledge acquisition system can be implemented, a means of providing or acquiring knowledge of the target system and its components is critical since without such knowledge, completeness can not be guaranteed. For example, the system must understand that LN2 flowing through a pipe has different characteristics than LN2 through a valve. A robust knowledge acquisition system must recognize these differences and the only way it can do this is by using the knowledge that the purpose of the valve is to redirect or change the flow of LN2. A more detailed treatment of remaining research to be done in automated knowledge acquisition is provided in PCA's final report to NASA [1].

The initial success of automated knowledge acquisition for model-based diagnostics is evidenced by the fact that NASA will be depending on KBDT to build its KATE diagnostic model of the tanking system. With continued development and enhancement, KBDT and future systems like it will become a standard part of the design repertory. Finally, it is clear to NASA that the historical design knowledge bases created by these automated knowledge acquisition systems will help to significantly improve the production and maintenance efforts for future systems. Therefore, NASA has decided to make automated knowledge acquisition a key component in all future NASA programs including the Space Station Freedom, the Advanced Launch System, and the National Aerospace Plane.

REFERENCES

1. Final Report: CAD/CAE Knowledge Base Development Tool, Prospective Computer Analysts, Inc., NASA contract NAS10-91496, January 1992. (Report data subject to SBIR rights.)

2. A Validation of Dependency Modeling and Testability Evaluation Using the Weapons System Testability Analyzer, Prospective Computer Analysts, Inc., Under Contract DAAB07-89-D-B805, Communications Electronics Command, Ft. Monmouth, New Jersey. February 1990.

3. NASA Kennedy Space Center, "KATE Knowledge Base Generation Guide". Internal Technical Report, NASA-KSC, Florida, November 1989.

4. Buttner, K., Sriram, D., and Freiling, M., "An Object-Oriented Blackboard Architecture for Model-Based Diagnostic Reasoning." Blackboard Architectures and Applications, New York, Academic Press. 1989.

5. Schank, R.C., Dynamic Memory, New York: Cambridge University Press, 1982.

6. Gonzalez, A., and Myler, H. Final Report: Automated Knowledge Generation, College of Engineering Technical Report 90-1623202/3. University of Central Florida, NASA Grant NAG10-0042. August 1990.

7. Selfridge, M. "Toward a Natural Language-Based Causal Model Acquisition System." Causal AI Models, New York, Hemisphere Publishing Corporation, 1990.

8. Horn, Werner., ed. Causal AI Models: Steps Toward Applications, New York: Taylor and Francis Group, 1990.

Figure 2