Issues in Automating the Extraction of a System Model from CAD Databases for Use in Model-Based Reasoning

AVELINO J. GONZALEZ
HARLEY R. MYLER
University of Central Florida

ABSTRACT: The knowledge acquisition task has proven to be one of the greatest obstacles to the widespread use of knowledge-based systems. Traditional methods for eliciting knowledge from the various sources (usually human experts) are generally labor-intensive. This has made development of such systems quite expensive in many cases. One attractive alternative for breaking the bottleneck is to automate the knowledge acquisition task.

Knowledge which is resident in many types of databases, such as in Computer-Aided Design (CAD) systems, provides an opportunity to fully automate the process of knowledge-base development. Diagnostic model-based reasoning systems represent a good opportunity for implementing this concept since their knowledge bases consist of a model of the system to be diagnosed (i.e., the "target system"). If the target system is drawn on a CAD medium, then much of the information required to compose this model is already contained in the CAD system database.

The information found in such a CAD database, however, is typically incomplete. Certain steps are necessary in order to generate the information required by a model-based system. This paper discusses the issues involved in developing a system capable of generating a model from a CAD representation of the target system. The resulting model serves as a knowledge base for a model-based diagnostic and control system. A brief description of a prototype capable of performing the above functions is also included in this article.

International Journal of Expert Systems Pp. 29–50, Volume 4, Number 1, 1991, ISSN 0894-9077.

1. INTRODUCTION

Ever since the knowledge acquisition bottleneck was recognized as a significant obstacle to further growth of knowledge-based systems (Feigenbaum, 1979) researchers have been investigating means of facilitating the process of developing knowledge-based systems. This research has yielded some interesting and novel approaches worthy of discussion.

The first and simplest approach has been to develop a methodology whereby the traditional knowledge acquisition process (i.e., personal interviews with experts) could be improved. The current emphasis is toward developing knowledge-based systems according to software engineering methodologies, thereby increasing the efficiency of knowledge engineering techniques. While these software engineering methodologies promise quick improvements in efficiency, the magnitude of the improvements is not expected to be large.

The second approach, generically called automated knowledge acquisition, can provide two kinds of assistance (Marcus, McDermott, & Wang, 1985):

- It can make it easy for someone with particular expertise to communicate that expertise.
- It can organize the knowledge that is communicated so that all of that which is relevant can be used to maximum effectiveness.

Knowledge elicitation systems (Parsaye, 1988) can automate the above process by carrying out a dialogue with the domain experts and then structuring the knowledge base in accordance with the expert's response. Because of the dynamic nature of this approach, such systems can facilitate the knowledge acquisition process tremendously.

Much research in this area stems from the work on Personal Construct Theory done by George Kelly (1955). Construct Theory proposes that humans attempt to categorize their personal experiences in such a way as to be able to predict events more effectively. Personal Construct Theory's major methodological tool, the repertory grid, was developed to elicit an individual's construct system, that is, "one's own finite system of cross-references between the personal observations he has made and the personal constructs he has erected" (Kelly, 1965). A repertory grid is essentially a complex sorting test in which a list of elements are judged successively on the basis of a set of bipolar constructs (Adams-Webber, 1987).

Knowledge engineering may be considered the process of making overt some part of an expert's personal construct system in the domain of interest (Shaw and Gaines, 1987). If it could be acquired, then the expert's unfiltered expertise could be elicited and represented as knowledge in a knowledge-based system. This would ameliorate the present obstacle of inarticulate experts.

The concepts of personal construct theory and repertory grids in particular, have been applied to knowledge acquisition by a number of researchers, PLANET (Shaw, 1985) ETS (Boose, 1984; 1985) and ICONKAT (Ford et al., 1991) among others.

However, not all knowledge acquisition tools are based on repertory grids. SALT (Marcus, McDermott, & Wang, 1985) assists in the knowledge acquisition for configuration tasks by using a problem-solving strategy consisting of generate,

test, backup, modify and regenerate. This approach guides the expert's interrogation, and is used to correctly represent the knowledge they provide. Other systems such as TEREISAS (Davis & Lenat, 1982) and MORE (Kahn, Nowlan, & McDermott, 1985) are intended for use in refining a knowledge base rather than for initial elicitation.

As knowledge-based systems become more common, the tendency may be for experts to attempt to develop their own (small) systems as a way to avoid the knowledge acquisition bottleneck. Tools such as those described above can be especially helpful to such experts who may lack the services of a knowledge engineer. For someone with limited experience in knowledge engineering, their use could make the difference between ultimate success of the knowledge-based system, and failure.

A third approach is to eliminate the expert knowledge acquisition process altogether. Model-based reasoning, whose main application is diagnosis of physical systems, basically accomplishes this by embedding the diagnostic knowledge within a generic algorithm. This algorithm is device-independent and it requires a model of the system to be diagnosed (the target system) in order to carry out its task. The required models are based on physical laws and not on expert opinion.

The knowledge in the reasoning mechanism can be likened to a test engineer who, armed with a measuring device and a system schematic, searches for a discrepancy in a system which he/she is troubleshooting (Hamscher & Davis, 1987). He/she then follows the flow of the medium (i.e., voltage, current, compressed air, etc.) upstream and downstream, making test measurements, until a faulty component is identified. This is, however, a simplification, because in typical monitoring systems not all components are remotely monitored. This can introduce a degree of uncertainty in accurately determining the single actual faulty component.

Model-based reasoning differs from the more traditional abductive reasoning systems in that it directly models the behavior of the device or system being diagnosed, rather than an expert's interpretation of the system's input/output characteristics. The latter typically uses rules to model the diagnostic expertise of one or more domain experts. This results in a shallow representation of the device where only events that have been previously observed by the expert are capable of being identified. Such is the basis for some of the earliest diagnostic systems, MYCIN (Buchanan & Shortliffe, 1985) and INTERNIST (Miller, Pople, & Myers, 1982). This approach has nevertheless proven highly successful in solving a variety of problems in diagnostics, as well as in such areas as design, planning, and classification. For cases where the device to be diagnosed cannot be adequately modelled, such an approach is clearly more advantageous than model-based reasoning.

Furthermore, abductive reasoning systems have been hindered by the fact that the knowledge needed to support domains of this type is often voluminous and complex, as well as quite difficult to elicit from available resources, mostly human experts (Hoffman, 1987). Two major reasons for this are that such knowledge is typically under-documented, and experts are often either unwilling or incapable of properly articulating their knowledge to a knowledge engineer. This may be because knowledge possessed by experts is often in a compiled form. That is,

the experts themselves may be unaware of how their knowledge evolved and why it is theoretically correct. They know only that it works because they have seen it work repeatedly throughout their years of experience. The knowledge elicitation tools described above address this problem.

Model-based reasoning systems, however, eliminate the knowledge acquisition problem altogether by embedding the generic diagnostic procedures in an algorithmic process that makes use of a model of the target system to identify possible causes of the failure. The reasoning algorithm itself is device-independent. It is the model description of a target system, however, that customizes the process to diagnosing one particular target system. Thus, the same model-based reasoning system can be used to diagnose different target systems by simply replacing the model. The model can be likened to a knowledge base, while the model-based reasoning algorithm can be likened to an inference engine in the more traditional abductive knowledge-based systems.

2. MODEL-BASED REASONING

The basis of the model-based reasoning technique is to reason about behavior of a system from the system's structure and function. (Diagnosis is a particular application of this.) More specifically, the model of a target system should describe the

- connectivity of its components
- functionality of each component.

Early work in model-based reasoning used combinatorial digital circuits as the domain for the model (Genesereth, 1984; de Kleer, 1979; Davis, 1984). DART (Genesereth, 1984) approaches the diagnostic task by generating tests that can be performed by a technician. The interpretation of the test results may then be used to determine the nature of the fault. DART makes no effort, however, to consider the cost of the tests suggested, or the diagnostic value of such tests.

The approach used by Davis (1984) was not to postulate possible fault mechanisms and then explore its consequences, but rather to look for a discrepancy and identify which component's failure could account for it. A discrepancy is a significant difference between the system's expected behavior and its actual behavior. The expected behavior can be simulated with the information contained in the model, (i.e., structure and function), and its input values. Input values are propagated through the system by the use of the constraints imposed on the components of the system. Actual behavior is measured through sensors, either manually or automatically. The basic idea behind this technique is to substitute hypothetical fault models by the violated expectations, and to use dependency records to trace back to the possible sources of a fault (Davis, 1984).

The work described above uses rules to express the functionality of the devices. Davis separates the rules into two types (simulation rule), which simulate the physical constraints of the device (e.g., the flow of electricity, etc), and inference rules, which represent conclusions we can make about the device. Rules are also used by another research system called HOIST (Whitehead & Roach, 1987). The rules in HOIST are referred to as causal rules.

Significant work in model-based reasoning has also been carried out by NASA at the Kennedy Space Center. Scarl et al. (1987) extends the above ideas to the on-line monitoring and diagnosis of process systems. Such systems are analog and, unlike the digital systems discussed above, are not combinatorial in nature. Additionally, it includes sensor failures in the set of candidate failed components. The Knowledge-based Autonomous Test Engineer (KATE) developed at Kennedy Space Center works by using a model of the target system to simulate its operation under the indicated conditions. The simulated system monitors itself at the same locations as the actual system, and the simulated sensor readings are compared to the actual sensor values. As long as the real system is operating normally, the parameters monitored should be in general agreement with those represented by the model (within a properly-chosen error band). Any disagreement between the actual sensor readings and the corresponding simulated parameters, however, causes a discrepancy to be noted.

KATE uses a Full Consistency algorithm to search for some fault that can explain what the sensors are showing (Scarl et al., 1987). It does not, however, have to test all components against all sensors because it bases its search on the original discrepancy (OD). The OD is defined as the first discrepancy noted after some (long) period of normal operation. A sensor's siblings are those other sensors which depend in any way upon the inputs that affect that sensor. A component's hypothetical value is that which describes what the state of the component is as implied by the observations made (i.e., sensor value).

In this algorithm, only those components which are upstream from the OD need to be considered possible causes of failure, or suspects. Instead of randomly generating hypothetical faults for the suspect, the OD is used to determine which faults are possible.

Upon detection of the OD, the KATE diagnoser is invoked and it locates all the suspects, the inputs, and the siblings of the OD. An attempt is made to determine a hypothetical value for each suspect based on the OD. Diagnosis is performed by taking all the suspect components and exonerating them until one (or a limited set) is found which can either be demonstrably declared the culprit, or which cannot be exonerated. A suspect is declared to be innocent, and thus exonerated if any one of the four criteria is true (Scarl et al., 1987):

- the suspect controls the OD only through components known to be innocent,
- no hypothetical value can be established for the suspect because the functional dependency of the OD cannot be inverted,
- the suspects hypothetical value agrees with its expected value, and
- the assumption that the suspect actually has its hypothetical value does not cause all sensors to become consistent.

Computing a hypothetical value means understanding what the measurement of the OD is telling us about the state of one of the suspects. The key process in KATE is the use of dependency inversion, where the functional representations of all the components can be inverted in order to calculate this hypothetical value.

KATE also has the ability to model non-local phenomena such as short circuits

```
(DEFRAME R3
    (NOMENCLATURE "RESISTOR R3")
    (AIO    RESISTOR)
    (SOURCE PW#3)
    (SOURCE-PATH CSTATUS-PW#3)
    (IN-PATH-OF   OPAMP#2)
    (UNITS "OHM")
    (STATUS  (/ (CSTATUS PW#3) (CSTATUS R3)))
    (CVALUE  150)
```

Figure 1: KATE Frame representation example.

in electrical networks, or flow in process systems by introducing a concept called psuedo-objects. Although rather informal in how they are implemented, these model non-existent components whose attributes contain system parameters and allow calculations which are global in nature, such as flow, resistance, pressure, temperature, etc. Pseudo-objects are used to form a network which does not necessarily correspond to the way in which the actual components are physically connected, but that interact with such a network to produce the proper simulation. This makes KATE able to handle problems significantly more complex than the combinatorial (local) problems solved by the researchers cited above.

When all of the components of a target system are connected, the properties of components up and downstream from a specific component will affect each component's transfer function. Thus, psuedo-objects attempt to resolve non-linear inconsistencies that arise within component sequence flows. Consequently, in order to add pseudo-objects to the knowledge base, the complete structure of the complex energy flows must be known, and some type of network analysis is needed.

KATE uses frames to represent its knowledge (model). Each component is represented by a frame, which is an instantiation of a higher level, more general component frame. The connectivity is expressed by slots describing the input and the outputs of each component. The functionality is likewise declared in a slot, as are other information such as the component description, its units, its range etc. Figure 1 shows a typical frame for the model-based system KATE (Cornell, 1986). The model acquisition system that is the subject of this paper is based on KATE.

By modeling the internals of the system to be diagnosed, model-based reasoning uses deeper reasoning since it is based on first principles. While it introduces some disadvantages over rule-based systems, model-based reasoning provides a powerful means of reasoning about the structure and behavior of a system.

3. MODEL CONSTRUCTION

Although model-based reasoning effectively eliminates the knowledge acquisition task (from experts), a target system model has to be developed nonetheless. This

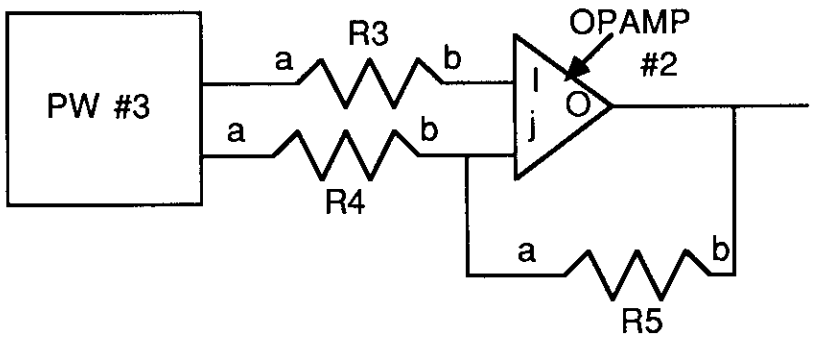

Figure 2: Schematic of simple electrical circuit.

is typically done from schematic diagrams of the target system by someone knowledgeable in the type of system to be monitored. Unfortunately, it can also be a time-consuming and error-prone task, since not all the required information is found in the drawings.

Over the last decade, a significant proportion of engineering design has been performed using Computer-Aided Design (CAD) Systems. CAD systems are sophisticated graphic software packages which are used to represent engineering design documentation such as drawings. The advantages introduced by such tools are many. Some of the most obvious are

- efficient filing and storage,
- increase in efficiency for the draftsperson,
- capability to zoom in on a part of a large drawing,
- 3-dimensional drawing capability,
- various options for surface presentation on drawing (e.g., wire frame, solid, etc).

Advanced CAD systems provide a database in which to store significant information about the drawing being represented. As an example of a drawing done on

Component Name	Component Description	Units
PW #3	Power Supply	VDC
OPAMP #2	OPERATIONAL AMPLIFIER	
R3	RESISTOR	OHM
R4	RESISTOR	OHM
R5	RESISTOR	OHM

Figure 3: Component list of circuit from Figure 2 in CAD database.

Component Name	Connect Point	Component Name	Connect Point
PW #3	+	R3	a
PW #3	-	R4	a
R3	b	OPAMP #2	I
R4	b	OPAMP #2	J
R4	b	R5	a
OPAMP #2	O	R5	b

Figure 4: List from CAD database showing component connectivity.

CAD, and the type of information that can be generated from the database, see Figures 2, 3, and 4. Figure 2 shows a schematic of a simple electrical circuit composed of a power supply (PW #3) connected to a operational amplifier (OP AMP #2) through three circuit elements (R3, R4, and R5). Figure 3 shows a list obtained from the database which describes the components found in the circuit drawing as defined by the draftsperson. Figure 4 shows another list which depicts the connectivity of the components, as well as their connect points (Myler et al., 1989).

Descriptions of physical systems on CAD invite the concept of extracting the information from the CAD database automatically, with minimal interaction with a human. By developing a system that analyzes a CAD database, and "generates" a knowledge base automatically, complete models (knowledge bases) could be developed in a fraction of the time presently required, even with the automated knowledge acquisition tools described above. The knowledge acquisition technique to be presented in this paper, unlike the others discussed in section 1, makes use of existing databases as the source of knowledge, rather than human experts. Thus, as a way of distinguishing it from those above, we will call it "Automated Knowledge Generation" (AKG), even though it is a bit of a misnomer because knowledge is not being generated; only the knowledge base is.

The knowledge contained in databases can be represented explicitly or implicitly. Explicit knowledge is where the knowledge is described as knowledge. One example is a textual representation of a troubleshooting manual in a database that contains a set of rules which describe a procedure or the heuristics to be followed by maintenance personnel in order to diagnose a piece of equipment. If a knowledge engineer were to read the document, he would be capable of representing this knowledge easily and almost directly, with little expertise about the domain.

Implicit knowledge, on the other hand, is that which is not explicitly represented as knowledge. Knowledge may have been used by a human (expert) to produce the contents of the database, but that knowledge is not explicitly set forth as such.

Extraction of implicit knowledge requires a certain amount of a priori knowledge about the domain on the part of the knowledge extraction entity (human or otherwise) in order to accurately extract it.

For example, in the case of physical systems, knowledge about the design of the system was required in order to carry out the design. The contents of the CAD drawing (and database), however, do not contain that knowledge explicitly. Thus, the knowledge to be extracted from the CAD database for the construction of a diagnostic model is implicit in the database.

4. PROBLEM DESCRIPTION

In the design of electronic or process systems which are typically represented by schematic drawings such as those of interest in this research, the available information from the CAD database normally includes a list of all the devices or components contained in the system. It appears to be a relatively simple task, therefore, to access the database in the CAD system and extract the information for the model from it. Generating a model would be largely a question of reformatting the description of the extracted model. Thus, a simple translator could presumably be built to convert the knowledge from the CAD database directly into a diagnostic frame-based model such as the one used by KATE.

Unfortunately, the situation is not quite this simple because the design information found in CAD is typically incomplete, and sometimes inaccurate. Using simple translation to generate a complete model is somewhat analogous to a third person walking into an on-going conversation between two other persons, thus being unaware of the context of the dialogue. With incomplete knowledge, the third person cannot meaningfully participate in the conversation. Without explicit complete information, a simple process of translation will not work.

There are various reasons for this lack of complete information in a CAD database description. Whereas the information contained in CAD system databases is generally sufficient to describe the structure of target systems rather well, the functional description of its devices has traditionally not been explicitly included in the drawings by its designers. It has generally been assumed in the engineering profession that anyone who needs to read the drawing will know what the components are designed to do. This leaves a large gap of knowledge necessary to build a model of a target system.

One way to overcome this lack of explicit information is to require that any new system or device designed on CAD include this information. This solution, however, precludes the generation of a diagnostic model from all existing systems that have already been designed in CAD up to this point. Moreover, since most draftspersons are not technically qualified to determine these parameters, that requirement poses a significant constraint in the efficiency of the design process.

Another reason for the incomplete or inaccurate nature of CAD drawing databases is that the drawing style of different designers on the CAD system may lead to a misinterpretation of the structure of the system. For example, Figure 5 depicts a representation of a single component (A) whose output feeds the inputs of three other components in parallel. A reasonably knowledgeable human would

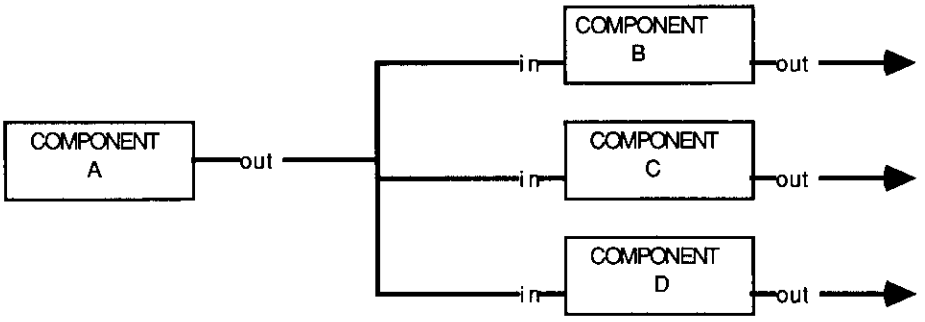

Figure 5: Component feeding three downstream components in parallel.

interpret this diagram correctly as described above. Nevertheless, cases have been discovered by the authors (Myler et al., 1989) where the connectivity has been interpreted by the CAD system to represent the drawing shown in Figure 6. This can be due to the sequence which the draftsperson uses to make the drawing. In the above case, it is likely that the designer (draftsperson) drew a line from the output of A to the inlet of B, and then drew lines connecting the inlet of B and the inlet of C to the line junction. Had the designer drawn it in any other sequence, the resulting interpretation might have been different.

Another problem arises when a measurement device taps into a line connecting two components, such as shown in Figure 7. In that schematic, the connection to the measurement should be considered an input to the latter, which has no output. Because the CAD system cannot discriminate between different components, the interpretation shown in Figure 8 can result.

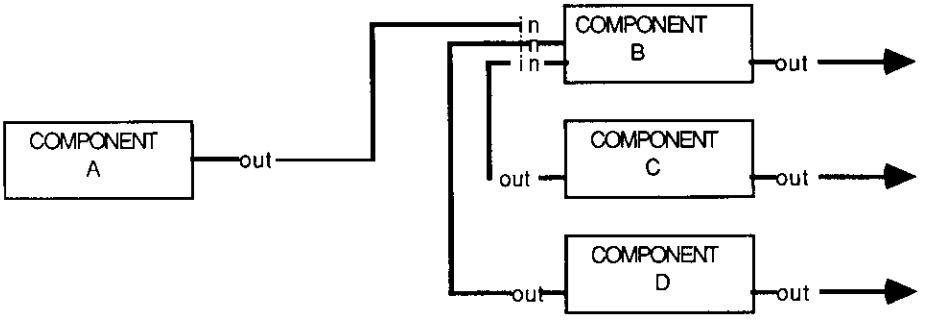

Figure 6: CAD interpretation of Figure 5.

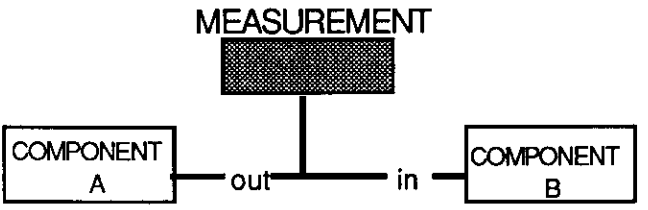

Figure 7: Sensor tap into process line.

One last reason is that the draftsperson, being human, is subject to errors in representing a system. Such errors can be simple such as misspelling words in the description of the system, or more serious ones such as incorrect connections of components. Errors can be especially common for inexperienced draftspersons.

The above problems tend to indicate that simple translation will not work. Fortunately, however, functional knowledge is generally represented implicitly in the CAD database, within the system design and the various component interconnections that are required to achieve the intended objective of the target system. A mechanism is therefore required that interprets the explicit knowledge in a CAD database in order to represent the correct structure and extract the implicit functional knowledge about a target system. This interpretation results in complete information about the target system of interest, which can then be used in the generation of a model for the latter.

The following two sections (sections 5 and 6) describe the interpretation process.

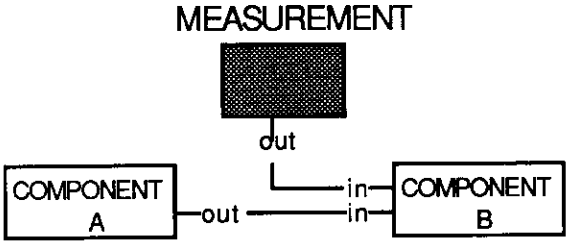

Figure 8: CAD interpretation of Figure 7.

5. AN INTELLIGENT INTERPRETER

The task thus faced by a system designed to generate a knowledge base (model) automatically from a CAD database is actually quite formidable. It must first extract the structural information from the CAD system, check it for consistency so as to avoid the misconnection problems described above, and then supply the functional knowledge which is not contained in the CAD database. A system such as this will be labelled an intelligent interpreter, in order to distinguish it from the simple translator described previously. The intelligent interpreter accomplishes the process of model generation within the contexts defined above by carrying out the following algorithm:

- Extract system connectivity from the CAD database.
- Identify each component with labels.
- Ensure consistency of the components' function and their connectivity.
- Endow each component with the appropriate functional attributes.
- Construct the frame-based knowledge structure.

The following sub-sections (5.1 through 5.5) describe the interpreter tasks required to generate a model of a system to be diagnosed. This model will be usable by a model-based reasoning system. The next section (section 6) briefly describes the AKG system developed by the authors which is capable of doing this.

5.1. Extraction of Connectivity

The process of extraction of connectivity is logically the first one to be attempted by the intelligent interpreter. It consists of two separate actions:

- accessing the CAD system directly and retrieving the information from it, and
- using that information to construct the basic structure of the model in terms of its connectivity.

The key issue here is that the interface with the CAD system must be designed to be generic enough so that, with minimum alterations, the interpreter can interface with, and access other types of CAD packages. The information can be extracted from the CAD database either directly or indirectly.

Direct extraction requires knowledge about the database being used by the CAD package in order to query it for the proper information. This approach reduces the genericity of the interpreter design.

The same information can be extracted indirectly through the use of macros within the CAD system database which generate the desired information in a standardized format for the interpreter. Although the obvious disadvantage to this is the need for manual coding within the CAD package, it is not unreasonable since these macros only have to be written once. Additionally, an engineering organization is likely to have a single CAD package in use throughout its various departments, thus keeping the amount of operator training and data sharing effort to a minimum. On the other hand, to design a truly generic interface module for

the interpreter would require a significant effort in an area that is not directly germane to the research being conducted.

This is the only sub-process that directly accesses the information contained in the CAD system database. Because of this, it should also be able to extract all other information which may be resident in the latter, and which can be useful to the interpreter later on. Examples of such information is the component's range of operation, its units and most importantly, its description, if any. The importance of the last two will become obvious in the next section.

5.2. Component Identification

As discussed above, the key requirement of the interpreter is that it be able to fill any gaps in the knowledge found in the CAD database. As discussed in section 4 above, the functional description of a component is typically not included in the drawing, and thus represents a large gap in the knowledge about the system to be modeled. The proper functional attributes can be assigned to the target system component by the interpreter only if the latter is correctly identified. For example, if a pump is incorrectly identified as a valve by the interpreter, then the functional attributes of the component would clearly be inadequate for diagnostic reasoning. The key, therefore, is to correctly identify each of the components in the target system. This task must be performed under the additional constraint of minimal human interface.

The task of identification assumes a base of a priori knowledge of components and systems in the relevant domain. Like an engineer looking over the schematic of a system, the intelligent interpreter needs to have such a priori knowledge about what typical components are and how they function. This internal knowledge base, called the Component Knowledge Base (CKB), stores the description of standard, generic components that are typically used in the type of systems to be diagnosed. Examples of such standard components found in the CKB include gate valves, solenoid valves, three-way solenoid valves, impeller-type pumps, compressors, switches, fuses, logic gates, power supplies, etc. Their description includes the functionality of the component, its units, its range, and the tolerance that they posses, among others. For a full description of the CKB, refer to McKenzie (1990).

The CKB is hierarchically organized, and it serves two purposes:

- aid in identification of the CAD components, and
- provide the missing information in the CAD database.

The main vehicle for identifying a component is its description, which is usually supplied in some form or another by the designer. Such descriptions range from the complete, correct and unabbreviated, to unreasonable facsimiles thereof. The task for the interpreter, therefore, is to parse through an unconstrained description, and match the key strings in the unknown device's description with those found in the CKB. A Heuristic String Matching Parser module within the interpreter (simply called the Parser) uses the description and the CKB to select possible matches of the unknown CAD component from the CKB (Kladke, 1989). It also ranks the possible matches with confidence factors, so as to indicate the most

likely identification. The Parser does this by

- using string-matching heuristics to parse through the description and isolate the meaningful information contained therein;
- using the meaningful information found in the previous step and comparing it to keywords found in the various levels of the CKB to select the best possible matches;
- further filtering the list of possible matches by looking for a consistency of units between the unknown CAD component and the possible matches. This will ensure that the possible matches in the component knowledge base are of the same type as the unknown device.

For a further discussion of the Parser module, please refer to Kladke (1989).

5.3. Resolution of Inconsistencies

This step is the most important one, and the one which separates this interpreter from a translator, because it looks at the system as a whole, making certain that all the components are properly identified in a way which makes them consistent with each other.

The Parser module will select a (hopefully) small list of possible identities of the unknown object, but the final identification of the device is not yet complete. The interpreter must still determine the single label most appropriate to the unknown device. It does this by comparing each of the possible matches to the unknown component's available information to ensure compatibility with its neighboring components as well as with the entire system. This compatibility depends on the unknown component's connectivity. This determination will be made after a constraint satisfaction process is carried out on the unknown component and its neighbors by the Resolver module of the interpreter.

Resolution of inconsistencies is the most critical sub-process of all because it must ensure that the entire model is consistent with itself. In addition, it must determine the final confidence placed on the identity and connectivity of each of the components in the target system.

The approach taken by the interpreter for resolution of inconsistencies in the connectivity and/or final component identification is to use constraint satisfaction.

One piece of information contained in the CKB for each of the generic components found therein is its constraints. These constraints define where and how this type of component would fit in within a typical system. The constraints include, but are not limited to

- the number of inputs allowed (minimum and maximum),
- the type of inputs allowed (e.g., voltage, current, flow, etc.),
- the general type of systems in which this component would be found (e.g., electrical, chemical process, etc.),
- the general environment in which this component is found (e.g., the nature of the surrounding components), and
- the type of compatible neighbors.

The constraints are divided into four categories: strong, medium, weak, and supporting. All the strong constraints listed in the CKB for a candidate identification must be satisfied by the unknown component in order for a candidate identification to remain in contention. They cannot be relaxed. Normal constraints, on the other hand, can be relaxed in order not to over-constrain the system. Weak constraints can be more easily relaxed, and supporting constraints are typically only examined when all else is equal in a comparison of two candidate identifications. Successful identification of an unknown component is determined by an evaluation of how well the unknown component satisfies the constraints of each of the candidate identifications. Conflicts arise when a component is able to satisfy only some of the non-strong constraints for more than one candidate identification.

The resolution of such conflicts involves looking for components whose identity is well-known compared with its surrounding elements in the system. These "islands of certainty" represent a starting point from which to begin to propagate the constraint satisfaction process. The resolver checks for

- node consistency,
- arc consistency, and
- path consistency.

Node consistency uses the strong constraints obtained from the possible matches and the known independent properties of the unknown CAD device such as units, range, etc., (which are extracted from the CAD database), and checks for consistency among them. Lack of satisfaction of a strong constraint eliminates the candidate match from further consideration, thus reducing the search space.

All remaining candidates then go through an arc consistency determination, which begins with the islands of certainty and looks towards their neighboring components to try to satisfy some of their strong constraints. Those whose strong constraints are not satisfied in the arc consistency test are also eliminated from further consideration.

The difference between arc consistency and node consistency is that node consistency only deals with those constraints which affect the unknown component itself. The arc consistency test, however, takes into account those others which relate to the unknown component's neighbors. This step will further eliminate candidate matches. If a candidate match is found for which all of its constraints at all levels (i.e., strong, medium, and weak) are satisfied, then that becomes the identity of the unknown component. If more than one component is found which satisfies this requirement, then the related constraints are used to break a tie. Failing that, the interpreter will resort to the human to indicate the true nature of the device. Once a component is resolved, (i.e., its identity determined) it then becomes an island of certainty and can be used to resolve other unknown components in the system.

Path consistency is the last of the functions performed by the Resolver. It identifies all the possible connected paths in the target system. All of the components in each path are checked to see whether they now represent islands of certainty. If all components in a path are islands of certainty, then that path is

considered to be consistent. Otherwise, arc consistency will once again begin with those that are not.

The goal is to increase the confidence factor of the identity of each of the unknown CAD components above a pre-set threshold. This indicates that identification has been achieved. If the confidence factor of one or more components cannot be brought above this level, then the human is queried as to its identity.

If the identity declared by the human is represented in the CKB, then that identity will be noted and the interpreter will continue to the next phase. Otherwise, the interpreter further queries the user for additional information, and then adds that generic component entry to the CKB. This crude sort of learning mechanism assures the continued growth of the CKB.

For a complete description of the Resolver module refer to Myler et al. (1989) and Towhidnejad (1990).

5.4. Functional Attribute Endowment

Once the identification of each component is verified in a system-wide context, then it remains for the functional description of each of the components to be retrieved from the CKB and incorporated with the rest of the information which makes up the model. This is an uncomplicated, yet distinct and necessary procedure which must be carried out in order to complete the model and compensate for the functional information missing from the CAD representation.

5.5. Construction of Frame-Based Knowledge

As a final step in the interpreting process, a model has to be built in the format required by the model-based reasoning system of choice. Models of target systems can be expressed quite differently in the various model-based reasoning systems available. Therefore, a significant aspect of the model construction process is that in a generic interpreter, the ability must exist to build a model for a variety of different model-based reasoning tools. This is not seen as a major problem, as long as the formats for the various tools are made available to the interpreter. It is simply a question of choosing from the various formats that may be predetermined for each of the different tools. Although the design and implementation of this module is also rather uncomplicated, its presence is important.

The informality of the method by which the pseudo-objects are created and added to KATE precluded the inclusion of the capability to model non-local behavior. Thus, the interpreter described is limited to a combinatorial type of system. Nevertheless, it is not seen as an insurmountable task to provide the interpreter with the ability to arrive at this type of equation. The inclusion of system flow knowledge, however, remains a topic for future research.

6. THE AUTOMATED KNOWLEDGE GENERATION SYSTEM (AKG)

Research in the generation of knowledge bases from CAD databases has been funded by NASA-Kennedy Space Center since 1987. Its goal has been to produce an intelligent system which will generate a knowledge base for a model-based

system given the CAD representation of an engineered system and a unconstrained CAD database related to the same system. Called the Automated Knowledge Generation System (AKG), it is based on object-oriented programming, and can be broken into eight major modules managed through a window-driven user interface:

- ACCESS, a data communication module for accessing CAD files residing in remote computers.

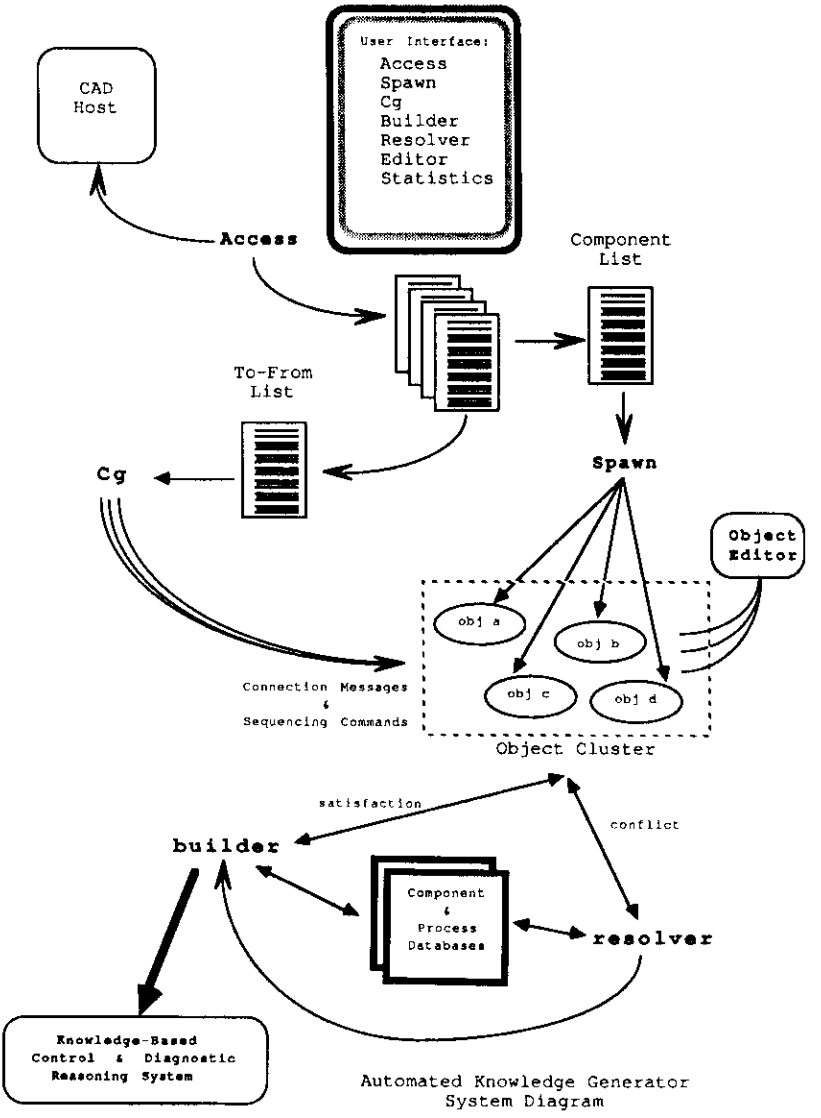

Figure 9: Automated knowledge generation system diagram.

- SPAWN, a module for creating Lisp objects (Flavors) from CAD component information.
- CONNECTIVITY GENERATOR, a module for interpreting (and if necessary, correcting) the component connectivity information from CAD.
- COMPONENT KNOWLEDGE BASE (CKB), a dynamic storehouse of descriptive and functional component and system information.
- PARSER, a component identification module responsible for providing a link between the raw, unconstrained descriptive information from CAD and the Component Knowledge Base.
- RESOLVER, a module for resolving conflicts which arise from incomplete or inconsistent data.
- BUILDER, a knowledge base frame builder.

These modules basically correspond to the functions described in section 5 above. See Figure 9 for a description of the system architecture. Detailed description of the system can be found in Gonzalez et al. (1989).

The research focused on CAD databases extracted from an Intergraph CAD representation of target process systems. The schematic drawing of the target system is typical of most drawings for similar process control systems. The research has been conducted on a Symbolics 3640 LISP machine. The majority of the generic components described in the CKB are for process systems. At the time of this writing, it includes approximately 400 such components.

7. TESTING AND EVALUATION OF AKG

AKG has been tested extensively on small, hand-coded systems as well as with CAD descriptions of real process systems found at the Kennedy Space Center. Tests performed on three actual process systems at the Kennedy Space Center represent the most significant part of the testing of AKG, and they are described in the following section.

The first test was performed on a small process system designed by NASA personnel to demonstrate the KATE model-based reasoning tool. Called the Purge Demo system, it consists of two small air compressors connected to a single pipe through a two-position solenoid valve, whose status is controlled by a simple control system overseen by KATE. This valve connects one or both of the compressors to a purge valve which exhausts the compressed air to atmosphere. The purge valve is also a solenoid valve controlled by KATE. Sensing devices are located at various places to monitor the performance of the system. KATE will turn on the second compressor if it detects a fault on the line which causes the air pressure to decay when the purge valve is open. The Purge Demo consists of 46 components, including power supplies, compressors, fuses, relays, logic gates, and valves. The CKB was populated with the devices found in the system as well as others that were not. The appropriate generic constraints were included for each of the generic components included.

Of the 46 components in Purge Demo, AKG correctly identified 33 of these

(72%). The 13 unresolved ones can be segregated into three groups, depending on the reason why they were not correctly identified. These are as follows:

Category 1: There was no generic component in the CKB which actually represented the unknown component (i.e., "by-pass glass"). Thus, no candidate label was found for the unknown component during the parsing operation. This is, of course, equivalent to a human not being familiar with the unknown component.

Category 2: The unknown component did not satisfy any of the strong constraints contained in the CKB for the candidate labels that were found during the parsing operation.

Category 3: Various competing labels remained for the same unknown component after the arc consistency test, and further discrimination between them was impossible.

For the Purge Demo system, 7 of the 13 unresolved components fell under category 1. It could be reasonably argued that a more robust CKB would have allowed these unresolved components to be identified.

None of the unresolved components fell under the second category, while the remaining six were due to category 3. It could likewise be argued that additional constraints in the generic component of the CKB would have allowed further discrimination between competing labels.

Although AKG was found to operate satisfactorily on the Purge Demo, it was decided by the research team that a more realistic test would be an actual process control system in operation at the Kennedy Space Center. The target system chosen for this evaluation is called the Environmental Control System in the Orbiter Maintenance and Refurbishment Facility (ECS-OMRF). It provided a more severe test of the functionality of AKG. The ECS-OMRF is composed of 133 components. It was during this test that the mis-interpretations of the system connectivities reported above were discovered. This required some adjustments in the operation of the Resolver module in order to address this problem.

The tests on the ECS-OMRF produced 114 resolved components (86%). Of the remaining 24, they were segregated as follows:

Category 1- 2,

Category 2-10,

Category 3-12.

Once again, the situation with the unresolved components in categories 1 and 3 could be corrected by a more robust CKB. The reason for the surprising number of unresolved components in category 2 was that one component in the CAD drawing described as a "remote temperature electric transducer" was actually a temperature control assembly composed of a silicon-controlled rectifier (SCR), a circuit breaker, and temperature controller, and a temperature transducer. In fact, the Parser passed the remote temperature transducer as a candidate label to the Resolver, but its constraints were not consistent with the unknown component's (the assembly) upstream and downstream neighbors, and it was thus rejected by

the Resolver. Although this points to a deficiency in AKG in that it was not able to detect an error in the drawing, considerable consolation exists in the fact that it did not incorrectly identify the assembly as a simple transducer, as it was erroneously described in the CAD database.

Finally, a third test was performed on a scale model of a tanking system which transfers liquid oxygen (simulated with water) from a main storage tank at the space center to the orbiter tank. This scale model system, called the "Water Tanking System," is made up of 71 components. This problem represented the stiffest challenge yet because it did not represent a completed design, but rather a draft. As such, there were several inconsistencies in the CAD database. For example, some of the components that were described in the component file were not connected to any other components in the system drawing. It was suspected that the database file did not correspond to the same version of the hardcopy drawing received. Nevertheless, for these reasons, it provided an interesting test of the capability of AKG under very unreliable data.

Not unexpectedly, only 33 of the 71 components were correctly resolved (47%). The breakdown for the unresolved components was as follows:

Category 1-17,
Category 2- 8,
Category 3-13.

It was encouraging, nevertheless, that AKG was able to resolve almost half of the components in a clearly corrupted database simply by understanding the relationships between components.

8. CONCLUSIONS

The ability to automatically retrieve and interpret diagnostic knowledge from a CAD database with minimum human intervention promises to relieve the knowledge acquisition bottleneck for a number of applications. The research described here indicates that this is possible. The key features of such a system are

- the ability to extract the connectivity from a CAD system database,
- the availability of a priori information about generic components to be used in target systems which can supply needed information typically not found in CAD system such as component functionality,
- the ability to identify the nature of all of the components defined by the CAD system database,
- the ability to use constraints to represent knowledge about the target system and employ these constraints to ensure system consistency and resolve all conflicts.

The research described in this paper has shown that the above features can be implemented in a system which successfully performs its intended task of extracting from a CAD database the implicit knowledge about structure and functionality of a physical system, for the purpose of autonomously generating a model of a

target system. The robustness of this Automated Knowledge Generation system, however, is directly related to the robustness of the Component Knowledge Base and the validity of the constraint hierarchy. Any efforts toward the improvement of AKG should be partially directed towards populating the CKB further. Additionally, the capability of handling global concepts such as system flow, as implemented in pseudo-objects in KATE also needs to be incorporated.

REFERENCES

Adams-Webber, J.R. (1987). Personal construct theory. In R. Corsini (Ed.), *Concise Encyclopedia of Psychology* (pp. 824–825). New York: Wiley Interscience.

Boose, J.H. (1984). Personal construct theory and the transfer of human expertise. *Proceedings of the National Conference on Artificial Intelligence.*

Boose, J.H. (1985). A knowledge acquisition program for expert systems based on personal construct psychology. *International Journal of Man-Machine Studies, 23*(5).

Buchanan, B.G. & Shortliffe, E.H. (1985). *Rule-Based Expert Systems.* Reading, MA: Addison-Wesley.

Cornell, M. (1986). Knowledge-based automatic test equipment. *Proceedings of the ROBEXS 86 Conference.* NASA Johnson Space Center.

Davis, R. (1984). Diagnostic reasoning based on structure and behavior. *Artificial Intelligence, 24,* 347–410.

Davis, R. & Lenat, D.B. (1982). *Knowledge-Based Systems in Artificial Intelligence.* New York: McGraw-Hill.

de Kleer, J. (1979). Causal and teleological reasoning in circuit recognition. *12TR-529, Artificial Intelligence Laboratory.* Cambridge, MA: MIT.

Feigenbaum, E. (1979). Themes and case studies of knowledge engineering. In D. Michie (Ed.), *Expert Systems in the Microelectric Age.* Edinburgh, Scotland: Edinburgh University Press.

Ford, K.M., Stahl, H., Adams-Webber, J.R., Cañas, A.J., Novak, J., & Jones, J.C. (1991). ICONKAT: An Integrated constructivist knowledge acquisition tool. *Knowledge Acquisition, 3,* 215–236.

Genesereth, M.R. (1984). The use of design descriptions in automated diagnosis. *Artificial Intelligence, 24,* 411–436.

Gonzalez, A.J., Myler, H.R., Owen, B.C., & Towhidnejad, M. (1989). Automated generation of knowledge from CAD design data bases. *Advances in Artificial Intelligence* (Volume 1). Greenwich, CT: JAI Press.

Hamscher, W. & Davis, R. (1987). Issues in model-based troubleshooting. AI Memo 893. MIT Artificial Intelligence Laboratory, March.

Hoffman, R.R. (1987). The problem of extracting the knowledge of experts from the perspective of experimental psychology. *AI Magazine,* Summer.

Kahn, G., Nowlan, S., & McDermott, J. (1985). MORE: An intelligent knowledge acquisition tool. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (Volume 1, pp. 581–584). Los Angeles, CA.

Kelly, G.A. (1955). *The Psychology of Personal Constructs.* New York: Norton.

Kelly, G.A. (1965). The role of classification in personality theory. In B. Maher (Ed.), *Clinical Psychology and Personality* (pp. 289–300). New York: Wiley.

Kladke, R.R. (1989). *A mega-heuristic approach to the problem of component identification in automated knowledge generation.* Master's Thesis. Department of Computer Engineering, University of Central Florida. Orlando, FL, December.

Marcus, S., McDermott, J., & Wang, T. (1985). Knowledge acquisition for constructive systems. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (Volume 1, pp. 637–639). Los Angeles, CA.

McKenzie, F.D. (1990). *The use of constraints to represent system knowledge in automated knowledge generation.* Master's Thesis. Department of Computer Engineering, University of Central Florida. Orlando, FL, April.

Miller, R.A., Pople, H.E. & Myers, J.D. (1982). INTERNIST-I: An experimental computer-based consultant for general internal medicine. *New England Journal of Medicine, 307*(8), 468–476.

Myler, H.R., Gonzalez, A.J., Towhidnejad, M., McKenzie, F.D. & Kladke, R.R. (1989). Automated knowledge generation from incomplete CAD data: Research results. In *Proceedings of the Second Florida Artificial Intelligence Research Symposium* (pp. 10–15). Orlando, FL.

Myler, H.R., Gonzalez, A.J., Towhidnejad, M., McKenzie, F.D., & Kladke, R.R. (1990). *NASA Year 2 Technical Report.* Department of Computer Engineering, University of Central Florida. Orlando, FL.

Parsaye, K. (1988). Acquiring and verifying knowledge automatically. *AI Expert,* May, 48–63.

Scarl, E.A., Jamieson, J.R., & DeLaune, C.I. (1987). Diagnosis and sensor validation through knowledge of structure and function. *IEEE Transactions on Systems, Man and Cybernetics, 11EMC-17*(3), 361–367.

Shaw, M.L.G. (1985). PLANET: Some experience in creating an integrated system for repertory grid application on a microcomputer. *International Journal of Man-Machine Studies, 23*(5).

Shaw, M.L.G. & Gaines, B.R. (1987). An interactive knowledge elicitation technique using personal construct technology. In *Knowledge Acquisition for Expert Systems* (pp. 109–136). New York: Plenum.

Towhidnejad, M. (1990). *Functional conflict resolution in automated knowledge generation.* Doctoral Dissertation. Department of Computer Engineering, University of Central Florida. Orlando, FL.

Whitehead, J.D. & Roach, J.W. (1987). Expert systems without an expert: Fault diagnosis based on causal reasoning. *Texas Instruments Technical Journal,* Winter.

Manuscript received October 1990; accepted December 1990.

Address all correspondence to Avelino J. Gonzalez, Department of Computer Engineering, University of Central Florida, Orlando, FL 32816-0950.