developing computerized systems in many domains. Such systems can vary in size from small-scale to very large scale systems.

One promising area for further research is to extend the MOPA concept to states. As the formalism now stands, the concept is used to organize activities only. Although states are linked to activities by the start and caused states, they are separate concepts. One can then have abstract states with the "has-states" slot pointing to a list of states known at that level. Having state abstractions will result in a more rigorous formalism. Another area of potentially interesting research is to extend AMS so that it has the capability to generate plans, store them, and execute them when it is desirable to do so. The plan and execution phases are subject to time constraints and can be done in a dynamic and incremental way, because it is not realistic to plan in detail a task that will be executed in three months.

ACKNOWLEDGMENT

We wish to thank N. Naffah and M. Ader of Bull S.A., and C. Ellis of the University of Colorado, for their constructive comments.

REFERENCES

[1] M. Ader, "The dynamic model," in *Intelligent Workstation*, Bull Tech. Rep., 1989.
[2] J. F. Allen and C. R. Perrault, "Analyzing intentions in utterances," *Artificial Intell.*, vol. 15, pp. 143–178, 1980.
[3] R. Alterman, "An adaptive planner," *Proc. AAAI-86*, vol. 1, 1986, pp. 65–69.
[4] ——, "Adaptive planning," *Cognitive Sci.*, vol. 12, pp. 393–421, 1988.
[5] J. S. Anderson and A. M. Farley, "Plan abstraction based on operator generalization," *Proc. AAAI-88*, pp. 100–104, 1988.
[6] W. Brauer, W. Reisig, and G. Rozenberg, Eds., *Petri Nets: Central Models and Their Properties*, pt. 1, Lecture Notes in Computer Science 254, 1986.
[7] ——, *Petri Nets: Applications and Relationships to Other Models of Concurrency*, Lecture Notes in Computer Science 255, 1986.
[8] S. K. Card, T. P. Moran, and A. Newell, *The Psychology of Human Computer Interaction*. Hilldale, NJ: Lawrence Erlbaum Associates, 1983.
[9] J. Chailloux, M. Devin, and J. M. Hullot, "Le-Lisp: A portable and efficient Lisp system," *ACM Symp. Lisp and Functional Programming*, Austin, TX, 1984.
[10] R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intell.*, vol. 2, pp. 189–208, 1971.
[11] P. E. Friedland and Y. Iwasaki, "The concept and implementation of skeletal plans," Tech. Rep. KSL 85-6, Dept. Comput. Sci., Stanford Univ., Stanford, CA, 1985.
[12] M. P. Georgeff and U. Bonollo, "Procedural expert system," *Proc. 8th IJCAI*, Karlsruhe, Germany, 1983.
[13] M. P. Georgeff, A. L. Lansky, and P. Bessiere, "A procedural logic," *Proc. 9th IJCAI*, Vol. 1, Los Angeles, CA, 1985.
[14] H. Kautz, and J. F. Allen, "Generalized plan recognition," *Proc. AAAI-86*, Vol. 1, 1986, pp. 32–37.
[15] A. M. Keuneke, "Device representation: the significance of functional knowledge," *IEEE Expert*, 1991, pp. 22–25.
[16] J. L. Kolodner, "Maintaining organization in a dynamic long-term memory," *Cognitive Sci. 7*, vol. 7, pp. 243–280, 1983.
[17] ——, "Reconstructive memory: A computer model," *Cognitive Sci.*, vol. 7, pp. 281–328, 1983.
[18] ——, *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model*. Hilldale, NJ: Lawrence Erlbaum Associates, 1984.
[19] J. Li, *AMS (Activity management system): Un systeme de coordination des activities bureautiques*, Ph.D. dissertation, de l'Ecole Nationale Supérieure des Télé communications, France, 1990.
[20] T. A. Linden, "Planning by transformational synthesis," *IEEE Expert*, Summer 1989, pp. 46–55.
[21] D. J. Litman and J. F. Allen, "A plan recognition model for subdialogues in conversations," *Cognitive Sci.*, vol. 11, pp. 163–200, 1987.
[22] J. L. Peterson, "Petri nets," *ACM Computing Surv.*, vol. 9, no. 3, Sept. 1977.
[23] ——, *Petri Nets: Theory and the Modeling of Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1981.
[24] W. Reisig, *Petri Nets: An Introduction*. New York: Springer-Verlag, 1985.
[25] E. D. Sacerdoti, "Planning in a hierarchy of abstraction spaces," *Artificial Intell.*, vol. 5, pp. 115–135, 1974.
[26] ——, *A Structure for Plans and Behaviour*. New York: Elsevier, 1977.
[27] R. C. Schank, "Language and memory," *Cognitive Sci.*, vol. 4, pp. 243–284, 1980.
[28] ——, "Reminding and memory organization : An introduction to MOPS," in W. Lehnert and M. Ringle, Eds., *Strategies for Natural Language Processing*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1982.
[29] M. Stefik, "Planning with constraints (MOLGEN: Part 1)," *Artificial Intell.*, vol. 16, pp. 111–140, 1981.
[30] ——, "Planning and metaplanning (MOLGEN: Part 2)," *Artificial Intell.*, vol. 16, pp. 141–169, 1981.
[31] J. Sussman, *A Computer Model of Skill Acquisition*. New York: Elsevier (North-Holland), 1975.
[32] J. Tenenberg, "Planning with abstraction," *Proc. AAAI-86*, Vol. 1, 1986, pp. 76–80.
[33] D. E. Wilkins, "Domain independent planning: representation and plan generation," *Artifical Intell.*, vol. 22, pp. 269–301, 1984.
[34] ——, "Hierarchical planning: Definition and implementation," *Proc. 7th ECAI*, 1986, pp. 486–478.
[35] ——, *Practical Planning: Extending the Classical AI Planning Paradigm*. San Mateo, CA: Morgan Kaufmann, 1988.

Validation of an Automated System Model Generator

Avelino J. Gonzalez, Harley R. Myler, Frederic D. McKenzie, Massood Towhidnejad, and Robin R. Kladke

***Abstract*— Modeling and simulation have always been highly useful techniques to use when designing, analyzing, or diagnosing an engineered system. Development of an appropriate model, in terms of accuracy, granularity, and complexity, has typically been the burden of the designer, analyst, or troubleshooter. It would naturally be advantageous if a model could be developed automatically, with the user supplying only some final minor refinements. The fact that most modern systems are designed in a computer-aided design (CAD) environment makes this a realistic prospect, because much of the data necessary for the model is already in electronic form. This article outlines a system called the automated knowledge generator (AKG), which embodies techniques that automatically create a model of an engineered system directly from its CAD representation, and describes the extensive testing process followed in order to validate its performance.**

***Index Terms*—Automated knowledge acquisition, model-based reasoning, knowledge-based systems, constraint satisfaction, computer-aided design systems, simulation models**

Manuscript received April 3, 1991; revised November 25, 1991, June 23, 1992, and May 27, 1993.

A. J. Gonzalez, H. R. Myler, and F. D. McKenzie are with the Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL 32816 USA; e-mail: ajg@engr.ucf.edu.

M. Towhidnejad is with the Department of Aviation Computer Science, Embry-Riddle Aeronautical University, Daytona Beach, FL USA.

R. R. Kladke is with Denver Aerospace Division, Martin Marietta Corp., Denver, CO USA.

IEEE Log Number 9213313.

I. Introduction

The development of a model that adequately describes an engineered system to be designed, analyzed, or diagnosed has always been a rather burdensome, but nevertheless requisite, task in the modeling and simulation process. There is a significant advantage to be gained from the ability to automatically generate a simulation or diagnostic model of the system in question. It would eliminate the effort required for a human to manually create the model, and would result in a reduction of errors. The fact that most modern systems are designed in a computer-aided design (CAD) environment makes this a realistic prospect, because much of the data necessary for the model are already in electronic form. This article outlines a system called the automated knowledge generator (AKG), which embodies techniques that automatically create a *device-centered* model of a system directly from its CAD representation. This model would then be ready for use in a diagnostic task.

A device-centered model of a system generally requires that each of its components be defined in terms of its connectivity to other components (system structure), and its individual local behavior, that is, its input–output (I–O) characteristics (function). Such a model is typically employed in model-based diagnostic reasoning systems where the behavior of the model is compared to that of an actual device in real-time and any resulting discrepancies suggest a malfunction in the real system.

Whereas the information that describes the structure of the modeled system is generally available from the CAD representation and is comparatively easy to extract, such is not the case for the component behavior representation. This is due to the fact that the typical system schematic diagram does not contain a description of the function of the components. It is assumed that the reader of the diagram already knows what it is, or knows how to determine what it is through either context or reference, or does not need to know it. Therefore, the only practical solution is to obtain the functionality description from an external source of domain knowledge. Such knowledge is implemented in the AKG system as a *component knowledge base* (CKB).

The CKB is designed to be a comprehensive database that includes all (or nearly all) of the components that are commonly used in the domain and describes their significant attributes, such as their behavior, among others. In order to eliminate confusion, the components of the physical system and of the corresponding model are referred to in this concise paper as "components," whereas the entries in the CKB that represent components in the domain are referred to as "CKB elements," or simply "elements."

In short, the AKG system determines the model connectivity by extracting the information necessary from the CAD database and the behavioral representation of its components by finding the corresponding element in the CKB and obtaining that information from the CKB element. The last process requires that each component in the system be identified and uniquely matched with its corresponding element in the CKB, whereupon the attributes of the corresponding CKB element can be imparted to the component representation.

An unknown component's name and description, usually obtained from the CAD database, are quite important in its identification. The components represented in the CAD system typically contain a label (name), which is a symbol depicting its internal identification. This label could be a descriptive name, such as "solenoid-valve," or, more typically, a code name, such as "SV201," that could stand for solenoid valve #201. Such labels allow the field installer to correlate the actual device with its location in the schematic diagram. Additionally, CAD systems often associate with the component a string description that contains a more comprehensible description (for humans), such as "Solenoid valve for the control of makeup water to the cooling water tank." Although such a descriptive string can be used to assist the process of identifying a component found in the CAD database, the task is more difficult than it might initially appear, because these descriptive labels are not standardized. Moreover, they are generally unconstrained in how they describe the component, and, on occasion, have been found to be misspelled, incomplete, abbreviated, or, in some cases, missing altogether.

AKG makes use of two techniques to carry out the identification of the unknown components. The first is a string matching process that takes the string description of the unknown component in the CAD representation and finds the CKB element(s) whose names most closely resemble the string description. The string matcher uses the CKB as the search space for this task, and assigns each potential matching CKB element with a number indicating how close a match it is (a high number indicates a close match). All those matches that score above a user-determined threshold value will be returned as "possible matches" and become part of the possible-match slot of the object representing the unidentified component.

If the string matching process returns exactly one possible match, it is likely to be the correct CKB element representation for the component in question. With few exceptions, however, the string matching process will return more than one possible match. Therefore, further filtering must be done to unambiguously determine the true identity of the unknown component. For this purpose, a second technique is brought to bear that uses a constraint representation and satisfaction algorithm to eliminate possible matches whose constraints are not compatible with the unknown component's known neighbors.

The use of constraints in design environments has been previously reported in the literature [1]–[5]. Buchmann's work, for example, uses these constraints to enforce relations between components determined by their compatibility as well as the desired system design. AKG extends this concept by using constraints to identify the nature of the components themselves.

AKG was specifically designed to develop a diagnostic model for use in the *knowledge-based autonomous test engineer* (KATE) model-based diagnostic reasoning tool developed by the National Aeronautics and Space Administration (NASA) and presently in use at the Kennedy Space Center. KATE uses a device-centered approach [6] that requires that the model be described in terms of the local connections of its components as well as their functionality.

II. The Automated Knowledge Generator System

The AKG process can be divided into three phases:

1) the acquisition and interpretation of the contents of the CAD database,
2) the identification of the system components and their endowment with functional attributes, and
3) the final generation of a model.

These are described below.

A. Phase I

AKG begins by directly accessing the CAD database where the system representation resides. The system components are *spawned* as object constructs in an object-oriented LISP environment, and are symbolically connected together, as prescribed by the contents of the CAD database. This is done by filling in the values for the slots of the spawned object that define the connectivity of the various objects, each of which represents one system component. The following program modules are used in the above phase.

- *ACCESS* communicates with the computer hosting the CAD representation in order to access the CAD files.

- *SPAWN* uses the component identifiers from the CAD database to create unique component objects within the AKG environment.
- *CONNECTION GENERATOR* establishes the connectivity between component objects based on information supplied by the CAD system.

Although hardly trivial, the tasks carried out in Phase I do not represent major obstacles in the investigation. See Gonzalez [7] for a discussion of the difficulties faced in this task.

B. Phase II

This phase of the AKG process is its most important: that of identifying all components defined in the CAD representation, providing them with behavioral characteristics obtained from the CKB, and ensuring that the identifications are all unique, and consistent—globally as well as locally. The following are the major parts of the system.

- *HEURISTIC STRING IDENTIFICATION (HSI)* is the string matching process described earlier, which parses the CAD component descriptive label string in order to obtain candidate matches from the CKB. It uses distance measuring techniques to calculate the distance between two strings. This is described in Kladke [8] and in Gonzalez [9].
- *RESOLVER* is the conflict resolution process, which uses the constraints defined in the CKB for the candidate matches identified by the HSI to test for consistency with the constraints defined for the unknown component's neighbors. It is used to resolve any ambiguities passed on by the HSI. For further information on RESOLVER, see Towhidnejad [10], [11].
- *COMPONENT KNOWLEDGE BASE (CKB)* holds all the elements typically found in the domain, including their behavioral definition as well as their constraints. This module has been described above in detail. For further information, see McKenzie [12].

In this phase of the process, RESOLVER first groups together the components that are not yet sufficiently well identified and passes them to the HSI. A component is considered not to be sufficiently well identified when the value of some key slots, such as the function, have not been filled. It is important to note that at this point, these components are only known by their identifiers, which may be nothing more than a simple numbering scheme, as described in the previous sections.

The identification process begins by isolating a description of the component from the unconstrained string label supplied by the CAD system. The distance between this string and the names of CKB elements is then evaluated to determine a measure of closeness. This is done by the HSI, which searches through the CKB looking for candidate elements that closely resemble an unknown component's string description. The HSI employs heuristics to compute the distance between the unconstrained string label of the unknown component and the names of the various CKB elements. The CKB elements whose identifying strings exhibit a similarity to the label string of the unidentified component that is measured to be higher than a user-determined threshold are then chosen as candidate matches, and are placed in the possible-match slot of the spawned component object.

If there is no element in the CKB that provides a good candidate match for the target system component, its identity can be obtained by querying the user directly. This new knowledge is promptly "learned" by AKG in the sense that the new component name and description are added as new elements of the CKB if it did not exist before. If the user does not provide an input, then RESOLVER will proceed to find a set of CKB elements whose constraints make them compatible with the unknown component. RESOLVER will use only constraint information to carry this out.

It is more likely, however, that there will be more than one candidate identification whose similarity to the unknown component's string label surpasses this threshold value. In this situation, a true and unique match for the unknown component still needs to be determined. The appropriate system domain knowledge, represented as constraints, is used for this purpose.

Once possible matches are established, the RESOLVER uses path and node consistency to determine the validity of a possible match. The path consistency is a measure of the possibility that the connections between a component and its neighboring components are valid. The node consistency represents a measure of certainty that a component has been completely identified.

The constraints associated with each possible-match element returned by the HSI contain information that constrains the component to existing in only certain types of systems, being able to be connected to only certain components, and then only in a particular way. Such constraints are intended to limit the number of possible matches down to one. However, when the constraints can only limit the possible matches to more than one, then that part of the constraint network is said to be *underconstrained.* RESOLVER solves this by obtaining more information (generating tighter constraints) from a close-by system path, or from a set of paths that contain a complete enough (identified) component from which to start propagating constraints. A component that contains sufficient information about itself to be unambiguously identified, is called an *island of certainty,* and may be used as the seed to propagate constraints to unidentified connected components. RESOLVER also stiffens the constraints by including those of the children of the candidate CKB element. The children are then posed as possible matches.

When the constraints limit the possible matches to zero, this portion of the constraint network is deemed *overconstrained.* RESOLVER then relaxes some of the constraints in order to allow a potential identification label to surface. In this case, weaker constraints that would normally restrict some of the possible matches are overlooked, so the constraints involved are said to be relaxed. This cycle of propagating, stiffening, and relaxing continues until the identification confidence factors for each of the component in the system are above a certain threshold, or until no significant progress in resolution is detected.

Upon completion of the algorithm, the system components, once known only by identifiers and scanty information, have become completely identified, and their structure and function has been determined. The target system model now can be directly generated.

C. Phase III

The formal generation of the model itself is the third phase of the AKG process and the least complicated. It represents the output of the program. It consists of the following module.

- *BUILDER* merely maps the internal identified component network into a format of the user's choosing.

The Builder module can be modified so that models with vastly different formats can be generated, depending on the requirements of the simulation or model-based reasoning system that will make use of the resulting model.

III. Validation of AKG

The key to any engineering investigation, of course, is the determination of how successful the innovation developed was. Therefore, the evaluation performed on the AKG system was quite rigorous, extensive, and thorough. There were two phases to the AKG testing program:

1) testing of the modules individually with hand-generated data in order to validate their own process, done by using small, hand-coded systems that were not represented in CAD; and
2) testing the entire AKG system on operational process systems available on CAD.

These tests represent a significant effort in the validation of AKG, and they are described in this section. Additionally, the test results will be compared with other previously published results to allow conclusions to be reached about the expected usefulness of the concepts embodied in AKG.

A. Testing of Individual Modules

All of the modules described in the previous section were thoroughly tested prior to integration within the AKG system. Nevertheless, only the heuristic string identifier (HSI), the component knowledge base (CKB), and the RESOLVER were rigorously validated, because these represented the novel aspects of this investigation. The other modules were of a simpler nature, and therefore were checked only for programming bugs.

The CKB being basically a database, was validated manually to ensure the correctness of the elements therein. This was carried out in a qualitative fashion by allowing the other members of the research team to review the contents of the CKB and commenting on the correctness of the elements and their attributes. Since there are no tests to report as part of this process, the validation of the CKB is not discussed further. This section focuses on the validation effort for the HSI and the RESOLVER.

Validation of the Heuristic String Identifier Module: The HSI matching process makes use of the CKB in order to select elements that compare well with the CAD description of the subject component. The validation of the HSI consisted of two groups of tests. The first one encompassed testing the HSI by providing it with 103 artificially generated, unconstrained component description strings. These strings were specifically designed to contain potentially troublesome aspects that could rigorously test the megaheuristics that make up its distance-measuring algorithm [9]. They represented cases where the following conditions existed:

1) preservation of word order was significant;
2) word order was irrelevant;
3) gross misspellings were present;
4) abstract abbreviations existed which differed from those in the CKB;
5) the more-general-than relationship between system components and the CKB element was tested;
6) both conceptual similarity and dissimilarity in the component's units was evidenced;
7) verbose descriptions existed which tested the HSI's pruning abilities.

All 103 descriptive strings were successfully identified. The tests proved the HSI to be robust when dealing with these kinds of abnormal descriptions.

The second group of tests consisted of presenting to the HSI descriptions of components in actual process system drawings. The drawing used was the stator cooling water system (SCW) for a large electrical power generator, courtesy of Westinghouse Electric Corp.

The SCW system contained longer, nonstandard, more complex, and possibly more realistic descriptions of components than the artificially generated descriptions of the first set of tests discussed earlier. This provided a good test of the brittleness of the HSI.

The CKB was purposely populated with industry standard descriptions of all the components included in the SCW system. Thus, acceptable matches could be found for all SCW components during the testing. However, the CKB descriptions did not always agree with the description of the corresponding components as described in the SCW system drawing. In this test, all 109 components in the SCW drawing were properly identified by the HSI. By proper identification, it is meant that the actual element in the CKB corresponding to the unidentified component in the drawing was in the list of likely matches instantiated to the possible-matches slot of the unidentified item, and the correct identification was the one with the highest match confidence factor.

Validation of the RESOLVER Module: Tests were carried out on two simple electronic circuits by slightly modifying their representation in CAD in order to validate the robustness of RESOLVER. These are described below.

Comparator Circuit Tests: The first such circuit is a comparator circuit consisting of seven elements such as resistors and operational amplifiers [10], [11]. There were four separate tests performed on the comparator circuit.

In the first test, all the components were described correctly in the CAD database, although they were not necessarily spelled correctly. For example, the resistor $R1$ was described as a "retester." This was done so that the HSI would not be able to find an exact match with 100% confidence. Moreover, all the components had corresponding elements defined in the CKB. The RESOLVER was able to correctly identify all the elements in the circuit, including the misspelled one.

The second test consisted of a component being mislabeled in the CAD database. Operational amplifier OP1 was incorrectly described as "but-valve." The HSI, expectedly, returned "butterfly valve" as a possible match for component OP1, which, of course, is incorrect. RESOLVER, however, recognized that a butterfly valve did not satisfy the constraints imposed in the CKB and eliminated it as a possible match. Because no other element was forwarded by the HSI as a possible match, the RESOLVER proceeded to search all leaf-level elements in the CKB to find one that satisfied the constraints for OP1. As a result, operational amplifier, the correct choice, was found as the identity by the RESOLVER. This test shows the resiliency of the RESOLVER in finding the correct identity even when a component is mislabeled.

The objective of the third test was to force the RESOLVER to execute its stiffening process in order to eliminate competing labels. As mentioned above, this is done by including the constraints of the children of each competing element. This situation was forced by describing OP1 as a "four-input amplifier valve," which resulted in both "valve" and "amplifier," to be passed on by the HSI as possible matches. RESOLVER applied the constraints for the children of "valve" and "amplifier" in the CKB to the unknown component. The constraints of "operational amplifier," a child of "amplifier," satisfied all the requirements imposed by OP1's connectivity, whereas those of the children of "valve" did not. Thus, it correctly chose "operational amplifier" as the identification of OP1.

Lastly, a test was performed in which the description of R1 was removed altogether from the CAD database, and the RESOLVER was asked to come up with the best possible label for the component. Since there was no description, the HSI was not invoked at all. RESOLVER came up with "resistor," "inverter," and "diode," all elements in the CKB whose constraints were consistent with R1. Having a descriptive string would have allowed the HSI to eliminate the last two possibilities before they were returned to the RESOLVER.

Full-Adder Tests: A full-adder circuit, which is somewhat more complex than the preceding circuit, was the basis for the next set of tests [10]. It contains devices such as inverters and AND- and OR-gates.

The first test of this group was a simple one in that all components were described correctly in the CAD database (including the spelling). The objective was to test the RESOLVER in a more complex

circuit, as well as to establish a baseline from which to make modifications in the succeeding tests. The HSI correctly included the appropriate elements as possible matches for all the components, and the RESOLVER selected the correct one in all cases.

The second test represented a modification of the first test in which a "NAND-gate" element was added to the CKB, along with its child, a "two-input-NAND-gate." Meanwhile, the description of component AND1 in the CAD database was modified to read "two inpot AND gote." This forced the HSI to include both "2-input-AND-gate" and "2-input-NAND-gate" as possible matches. However, since neither of these had any children from whom to reverse-inherit constraints, the RESOLVER was not able to discriminate any further, and did not uniquely identify this component.

The third test changed the description of the AND1 component to "valve and gate." This resulted in both "valve" and "AND-gate" competing for the identity of AND1. RESOLVER executed the stiffening process and correctly selected "AND-gate" as the identity of AND1.

The fourth test included a description for AND1 as "four input butterfly valve and gate." Once again, the HSI returned "butterfly valve" and "4-input-AND-gate" as possible matches. Since AND1 does not have four inputs, the "4-input-AND-gate" possible match was not found to be consistent with AND1. Likewise, "butterfly valve" was not considered to be consistent and was eliminated. Having no other possible matches to consider, RESOLVER executed its relaxation procedure where the constraints of the parents of the possible matches are considered. It was found that the constraints of an "AND-gate" element (the parent of "4-input-AND-gate") were found to be consistent with AND1, and it was so selected as the unique identity.

B. Tests on Existing Systems

Tests were also carried out on three full systems that were used as testbeds [7]. The tests to be described had as their objective the automated generation of a KATE model (knowledge base) for each of the systems described. A KATE model consists of a set of frames, each representing a system component, with various slots, such as NOMENCLATURE, SOURCE–PATH (S–P) (inputs), IN-PATH-OF (I–P–O) (outputs), STATUS (functional description of an analog component in terms of the relationship between its inputs and outputs), DELAY, TOLERANCE (TOL), RANGE, and UNITS.

Generation of a KATE model consists of the filling in of the appropriate values for the slots in the frame representing each of the components in the system. It should be noted, however, that not all frames contain the same number or types of slots, because some may be unnecessary for the component that the frame represents.

The first testbed was the CAD representation of a small process control system designed and built by NASA to demonstrate the capabilities of KATE. Called the Purge–Demo System, it consists of 46 components, such as power supplies, compressors, fuses, relays, logic gates, and valves. For these tests, the CKB was populated with elements corresponding to all components found in the Purge–Demo System, in addition to others.

The second testbed was more realistic, because it represented a working process system at the Kennedy Space Center. The target system chosen for this test is the Environmental Control System in the Orbiter Maintenance and Refurbishment Facility, and is referred to as the ECS–OMRF. The ECS–OMRF is composed of 138 components. Its larger size and greater complexity provided a stiffer test of the validity of AKG.

The third testbed was a scale model of a tanking system that transfers liquid oxygen (LOX) from a storage tank at the space center grounds to the main tank of the orbiter. This scale model system is used to simulate the process and uses water instead of LOX. It is called the Water Tanking System (WTS), and it is made up of 71 components. This problem was particularly challenging for AKG, because it represented an incomplete draft of the design. Thus, several inconsistencies in the CAD database were found upon visual comparison of the contents of the database to the presumably corresponding hardcopy drawing furnished by NASA. For example, some components described in the CAD file were not connected to any other components in the system hard copy drawing. In light of this, it was suspected that the CAD database file received from NASA did not correspond to the same version of the hard copy drawing received. Although AKG does not make use of the hard copy drawing, it is a useful device for the investigators to determine whether the AKG results are correct. Nevertheless, for these reasons, it proved to be an interesting test of the capability of AKG when presented with unreliable data.

TABLE I
TEST RESULTS

Test name	Total components	Resolved correctly	Category 1	2	3
Purge-Demo	46	33	7	0	6
ECS–OMRF	138	114	2	10	12
Tanking system	71	33	17	8	13

The objective of these tests was to present AKG with the CAD database representation of a process system, and to determine how many of the components it was able to identify correctly (i.e., "resolve"). The reasons for the unresolved components were analyzed, and the results showed that causes of unresolved components could be segregated into three groups.

Category 1: No element existed in the CKB that corresponded to the unknown component.

Category 2: All of the strong constraints contained in each of the CKB elements listed as possible matches were not satisfied, causing all elements to be discarded. This represents an overconstrained condition that is beyond what AKG's relaxation technique can compensate for. It implies improper strong constraint specifications in the CKB element.

Category 3: More than one possible match remained for the same unknown component after the consistency test, and resolution of this ambiguity was impossible. This signifies an underconstrained situation beyond what AKG's stiffening process could compensate for, either because there were no islands of certainty near the unidentified component or because the constraints provided for the appropriate CKB element were either insufficient or incorrect. The results of the three tests are depicted in Table I.

For the Purge–Demo test, it was initially surprising that so many components fell under category 1, because the CKB had been purposely populated with all the components in that system ahead of time. The problem was found to be that the CAD system identified as system components the symbols used in drawings for the continuation of the drawing on another page (all seven). Because these, of course, were not represented in the CKB, they were not properly identified. It would not be expected that such components would be represented in the CKB.

The unresolved components of categories 1 and 3 in the ECS–OMRF test could be resolved by a more extensive CKB. For category 2, the reason for the surprisingly high number of unresolved components centered around one component in the CAD drawing that was incorrectly described in the CAD system as a

"remote temperature electric transducer." This device was in reality a temperature control assembly composed of a silicon-controlled rectifier (SCR), a circuit breaker, a temperature controller, and a temperature transducer. This represented 4 components out of the 10 found to be incorrect.

It was interesting to note that in the above case, the HSI passed the label "Remote-Temperature-Transducer" as a possible match to the RESOLVER, but its constraints were not consistent with the assembly's neighbors, and it was thus rejected. Although this points to a deficiency in AKG in that it was not able to detect an error in the drawing, some encouragement can be drawn from the fact that it was able to determine that the unknown component was not consistent with the simple transducer label erroneously given to it in the CAD database description. The other six remaining unresolved components found in category 2 were attributed to improper or incorrect constraints in the CKB.

In the Tanking system test, only 33 of the 71 components were correctly resolved (47%). This was not unexpected, and was largely due to the unreliability and the internal inconsistencies found in the draft version CAD representation provided. The CKB was populated with elements corresponding to those found in the hard copy drawing, which did not appear to correspond very well with the CAD database. This explains why so many components fell under category 1. Nevertheless, it was encouraging that AKG was able to resolve almost half of the components in a corrupted CAD representation by making use of process system knowledge to understand the possible relationships between the system components and to resolve any ambiguities.

In summary, provided with a different types of systems to analyze, the simple (Purge–Demo), the realistic (ECS–OMRF), and the difficult (Tanking System), AKG was able to correctly identify 72% of the components presented to it. It is estimated that with a more robust CKB, this number could be improved significantly.

C. Conclusions and Comparison of Results

The results of the tests above tend to indicate that the validity of AKG depends to a large extent on the competency and completeness of the CKB. Not surprisingly, the more elements contained in the CKB, the lower the chances that any unresolved component will fall under category 1. However, it also became obvious that the manner in which the CKB elements are represented within the CKB is almost as important as the number of elements found in it. The constraints representing the elements and how these constraints are classified (i.e., strong, weak, or normal) were found to be the most significant of the attributes of the CKB elements.

Further evaluation of the results of AKG was done by comparing its results to those obtained and reported by Thomas [13] while working on the Purge–Demo System using a simple translation technique that did not make use of the constraint propagation or heuristic string identification to automate the model generation task. Table II compares those results to the ones obtained by AKG.

As can be seen for the above table, the addition of an ability to reason about the makeup of the system, imparted through the heuristic string identification and the satisfaction of constraints, greatly improved the results obtained in previous work.

TABLE II
COMPARISON OF RESULTS

Attributes	Percentage of components whose attributes were correctly determined by the system: Translation results (%)	AKG results (%)
Nomenclature	0	100
Source-path (inputs)	15	100
In-path-of (outputs)	100	100
Tolerance	0	72
Delay	0	72
Status (function)	0	72
Units	100	100
Range	100	100

IV. SUMMARY AND CONCLUSION

The results of the testing described above show that the techniques employed in the AKG system provided significant advantages when compared against a simple translation scheme. Although it is not expected that AKG will generate the complete model in its final form, the authors believe that its use will greatly simplify the overall task of creating the model. Assuming an extensive component knowledge base, AKG will require more than token assistance from the user only in situations where the system contains little-known components, unrelated symbols in the drawing (such as the page continuation symbol), or very poorly or incorrectly labeled system components in the CAD representation. It is interesting to note, however, that in the case of the page continuation symbol, AKG did "resolve" the unknown component to a filter, even though there were no possible matches passed on by the string matching routine (HSI). In summary, it has been shown that using the techniques described above, AKG can adequately reason about the makeup of a system schematic diagram.

REFERENCES

[1] A. Buchmann, R. Carrera, and M. Vasquez-Galindo, "A generalized constraint and exception handler for an object-oriented CAD-DBMS," *Proc. IEEE 1st Int. Workshop Object-Oriented Database Syst.*, 1986, pp. 38–49.

[2] S. M. Ervin and M. D. Gross, "RoadLab: A constraint based laboratory for road design," in *Knowledge-Based Expert Systems in Engineering: Planning and Design*. Billerica, MA: Computational Mechanics, 1987, pp. 167–184.

[3] A. Borning, "The programming language aspects of the Thinglab, a constraint-oriented simulation laboratory," *ACM Trans. Programming Languages Syst.*, vol. 3, pp. 353–387, 1981.

[4] M. D. Gross, S. M. Ervin, J. A. Anderson, and A. Fleisher, "Constraints: knowledge representation in design," *Design Studies*, vol. 9, pp. 133–143, July 1988.

[5] D. Serano and D. Gossard, "Constraint management in conceptual design," in *Knowledge-Based Expert Systems in Engineering: Planning and Design*. Billerica, MA: Computational Mechanics, 1987, pp. 211–224.

[6] E. A. Scarl, J. R. Jamieson, and C. I. DeLaune, "Diagnosis and sensor validation through knowledge of structure and function," *IEEE Trans. Syst., Man, Cybernetics*, vol. SMC-17, no. 3, pp. 361–367, May–June 1987.

[7] A. J. Gonzalez and H. R. Myler, "Issues in the automated generation of a knowledge base from a CAD representation of a system," *Int. J. of Expert Syst.: Res. and Applic.*, vol. 4, pp. 29–50, 1991.

[8] R. R. Kladke, "A mega-heuristic approach to the problem of component identification in automated knowledge generation," M.S. thesis, Dept. of Comput. Eng., Univ. of Central Florida, 1989.

[9] A. J. Gonzalez, H. R. Myler, and R. R. Kladke, "Identification of unconstrained item descriptions using string-match heuristics," *Int. J. Expert Syst.: Res. and Applic.*, vol. 4, pp. 337–364, 1991.

[10] M. Towhidnejad, "Functional conflict resolution in automated knowledge generation," Ph.D. dissertation, Dept. of Comput. Eng., Univ. of Central Florida, 1990.

[11] M. Towhidnejad, H. R. Myler, and A. J. Gonzalez, "Constraint mechanism in automated knowledge generation," *Applied Artificial Intell.*, vol. 7, pp. 139–169, 1993.

[12] F. D. McKenzie, "The use of constraints to represent process system knowledge in automated knowledge generation," M.S. thesis, Dept. of Comput. Eng., Univ. of Central Florida, 1990.

[13] S. J. Thomas, "Automated construction of a knowledge base from computer-aided design data," NASA Kennedy Space Center Internal Tech. Rep., Artificial Intell. Sec., 1987.