N92-22685

# KNOWLEDGE-BASED AUTONOMOUS TEST ENGINEER (KATE)

**Carrie L. Parrish, Ph. D. and Barbara L. Brown**
**NASA Kennedy Space Center, DL-DSD-23**
**Kennedy Space Center, FL 32899**

## ABSTRACT

Mathematical models of system components have long been utilized to allow simulators to predict system behavior to various stimuli. Recent efforts to monitor, diagnose and control real-time systems using component models have experienced similar success. NASA at the Kennedy Space Center is continuing the development of a tool for implementing real-time knowledge-based diagnostic and control systems called KATE (Knowledge-based Autonomous Test Engineer). KATE is a model-based reasoning shell designed to provide autonomous control, monitoring, fault detection and diagnostics for complex engineering systems by applying its reasoning techniques to an exchangeable quantitative model describing the structure and function of the various system components and their systemic behavior.

## INTRODUCTION

Conventional approaches to developing and maintaining diagnostic and control process software result in a time-consuming and costly effort. Furthermore, the resulting software may be incomplete and unable to handle situations that were unforeseen when the software was written. Significant advantages are obtained by systematically representing and using knowledge of a physical system's structure and function to reason about its health and proper functioning, a technique referred to as "model-based reasoning". The Artificial Intelligence Section, Engineering Development Directorate, at the Kennedy Space Center, employs the concept of model-based reasoning in the development of real-time knowledge-based diagnostic and control systems for ground launch operations. The project resulting from these efforts is called KATE (Knowledge-based Autonomous Test Engineer). KATE is being designed as a generic, model-based expert system shell, incorporating the engineer's reasoning about control and diagnosis of complex engineering systems in the form of general software algorithms. KATE further embodies concepts of sensor-based and model-driven monitoring and fault-location and performs control and redundancy management of process control systems through application of the generic algorithms to an exchangeable knowledge base, which describes the structure and function (i.e., mathematical model) of a specific domain. Refer to Figure 1 for a pictorial representation of the KATE System.

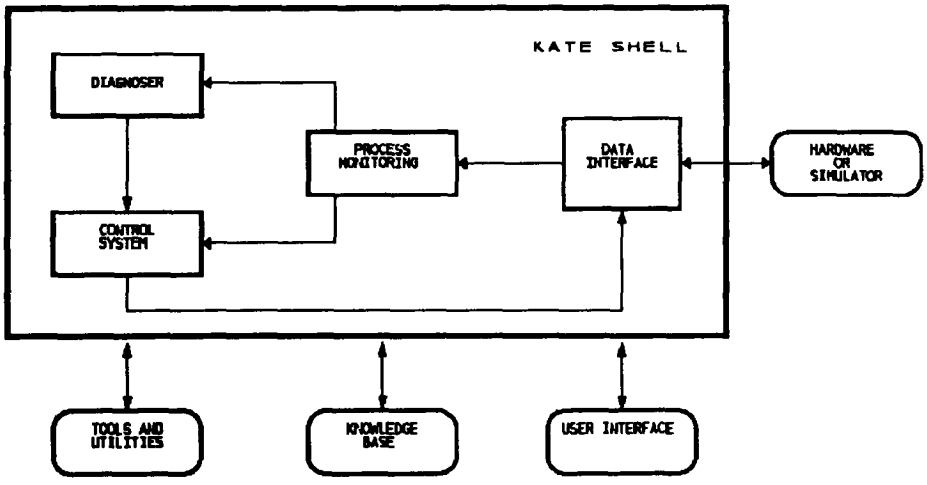


Figure 1. KATE Block Diagram

## BACKGROUND

NASA at the Kennedy Space Center, first developed LES (LOX Expert System) in 1983 to perform fault detection and isolation for the process control system which handles the loading of Liquid Oxygen into the external tank of the Space Shuttle. In 1985, LES grew into the more generic and robust model-based system, KATE.

During fiscal year 1989, KATE was successfully demonstrated against a scale model of the Orbiter Modification and Refurbishment Facility's (OMRF) Environmental Control System (ECS). The OMRF ECS supplies conditioned air to four different compartments of the Orbiter while the Shuttle is being processed in the OMRF. The hardware model of the OMRF ECS contains a purge unit that supplies chilled air to four ducts which, in turn, supply air to the Orbiter. Each duct consists of a heater for maintaining a constant temperature and a motorized flow-control valve for controlling the flow rate. A failure panel has been added to allow for manual failing of various components during testing. An operator can request certain flow and temperature setpoints and KATE will respond by adjusting the valves and heater output to achieve the setpoints. KATE will also identify and respond to external inputs to the ECS, such as load changes. In addition to control, KATE monitors the system, identifying discrepant measurement values and assigning probable causes. In most cases, control of the system can continue, especially if the discrepancy is determined to be due to a measurement failure.

In February of 1990 implementation of the KATE Generic Control System (GCS) prototype was successfully integrated into the Generic Checkout System. The Generic Checkout System is comprised of a network of UNIX-based equipment connected to a physical model of a Shuttle water tanking system, referred to as the Red Wagon. KATE-GCS performed control and diagnosis of fastfill, chilldown, and other processes related to filling the Red Wagon external tank.

## KATE KNOWLEDGE BASE

The knowledge base contains all information specific to the domain, i.e. the electro-pneumatic system being modeled. Generic component definitions as well as application specific data - classes of components, component names, tolerances, output-functions, inputs, delays, ranges, any information relevant to system operation, is stored here. The knowledge base can be thought of as a blue print of the system. When connected to hardware, the KATE shell utilizes this blue print to establish a nominal, working system model. This internal model can then be referred to as KATE monitors system health to detect anomalies, diagnose and recover from failures.

KATE's knowledge base has been developed through a combination of component modeling and mathematical modeling. That is, the knowledge base not only describes the system architecture, but is also the repository for functional information. System structure, or connectivity of system components, is represented primarily using data structures called **inputs** and **outputs** (described below). This information is used by KATE to discern control paths from command to measurement and is vital to gathering valid suspects during diagnosis of and recovery from failures. Mathematical equations of component output functions are also computed and stored in the knowledge base. Output functions describe how a component should behave based on its inputs. By representing structural and functional information, the dynamics of system's structure as well as its function can be made use of to perform monitoring, control and diagnosis. The knowledge base is separate from the KATE shell and therefore exchangeable. Conceivably, the shell might be used to monitor and control several hardware systems by the simple exchange of knowledge bases.

### Knowledge Representation

The knowledge base is based on a hierarchy of components and is separated into 3 major levels: the top, mid, and instance level. System components are modeled from as high level a concept as a "thing" in the top level, down to specific individual line replaceable units, such as a bypass valve being operated in the field, in the instance level. The top level of the knowledge base provides the framework, or building blocks, for system modeling. Here, broad concepts and classes of components are defined. The mid level houses generic definitions of system components - butterfly valves, transducers, relays, flow meters, schematic pages, level sensors, circuit breakers, and fuses to name a few. The instance level contains domain specific information, such as calibrations for system components that override mid level default parameters. For example, the mid level generic definition of a valve may contain default operating ranges for the valve. This default may be overridden in the instance level

definition of an actual occurrence of a butterfly valve in the field of operation. Inheritance of frame information is accomplished via AIO and AKO slots (described below) during interpretation at run-time. The use of component inheritance provides a means by which any component can be trace backwards to its top level definition (or visa versa) and reduces the amount of information that must be stored in the knowledge base.

This object-oriented approach to knowledge representation greatly increases the ease of adding or removing system components. Should it become necessary to do so, a component may be deleted or "disconnected" from the knowledge base, connection points re-established, and the model is once again complete without the need for costly recoding that is necessary with conventional software design.

Pseudo Objects and External Influences

In an electro-mechanical system there exist components that are not actual "physical" components, but rather, engineering concepts that manifest themselves during system operation. The effects of these "non-physical" components are observable, and can be controlled and diagnosed if necessary. Early KATE developers found it useful to model these **pseudo-objects**, as they are called in the KATE knowledge base, for the proper functioning of the KATE shell software. In general, pseudo-objects represent information that has no corresponding sensor for measurement, but can be computed from other data. Examples of pseudo objects are pressures, temperatures, flow rates, and tank levels.

Manual valves and orifices are examples of **external influences**. When all system components are functioning according to expectations, manual operation of an external influence component might explain anomalous system behavior. Pseudo-objects and external influence object definitions, like any other objects, utilize inheritance and are defined at all levels of the knowledge base in the same fashion as actual existing, line replaceable components.

Knowledge Base Structure

A frame representation language has been employed for data representation. Frames contain "slots" of information, and utilities have been constructed to allow for manipulation of these slots during knowledge base development. Slots in a frame may be thought of as records in a data structure. Each frame contains a description of the object and its connection to other objects. At the instance level a frame definition can be found for each hardware component or pseudo object in the system. This statement however, is not meant to mislead the reader; for certain types of components, other frame definitions may be necessary to complete the system model.

KATE currently makes use of forty odd slots for modeling purposes. These slots contain information that applies to icon display, component location for camera tracking, or other information necessary for the KATE shell to effectively perform monitoring, control, and diagnosis. Selected knowledge base slots are described here for the reader's use in a later example of knowledge base construction.

**NOMENCLATURE** - Text string denoting the frame's object name and description.

**AIO (An Instance Of)** - Indicates the generic object type from which the component will inherit characteristics. For example, if A75064 is a pressure transducer, the AIO slot in A75064's instance level frame definition would read (aio PRESSURE-TRANSDUCER).

**AKO (An Kind Of)** - Indicates the class type of the object. For example, the AKO slot in the mid level frame definition of PRESSURE-TRANSDUCER would read (ako TRANSDUCER).

**KINDS** - Inverse slot link to AKO. List of all frames in the knowledge base that represent a class of object. For example, the KINDS slot in the top level definition of TRANSDUCER would read (kinds PRESSURE-TRANSDUCER...).

**INSTANCES** - Inverse slot link to AIO. List of all component names of a certain generic type existing in the field. For example, the INSTANCES slot in the generic component frame definition of PRESSURE-TRANSDUCER would read (instances A75064 ...).

**INPUTS** - list of all frame names of the object's inputs.

**OUTPUTS** - List of all frame names of the object's outputs.

**CHANNEL** - Indicates the Hardware Interface Module, card, and position number from which the measurement/command is polled/commanded.

**COMPONENT-OF** - Indicates a component's Hardware Interface Module and card number.

**COMPONENT-POSITION** - Indicates the position of the measurement or command on the polling card.

**PHYSICAL-LOCATION** - Text string denoting the components location in the field.

To better illustrate the method used to construct KATE's knowledge base from system schematics, a sample drawing, taken from an advanced electrical schematic is used (Figure 2). This drawing depicts a 3-way solenoid

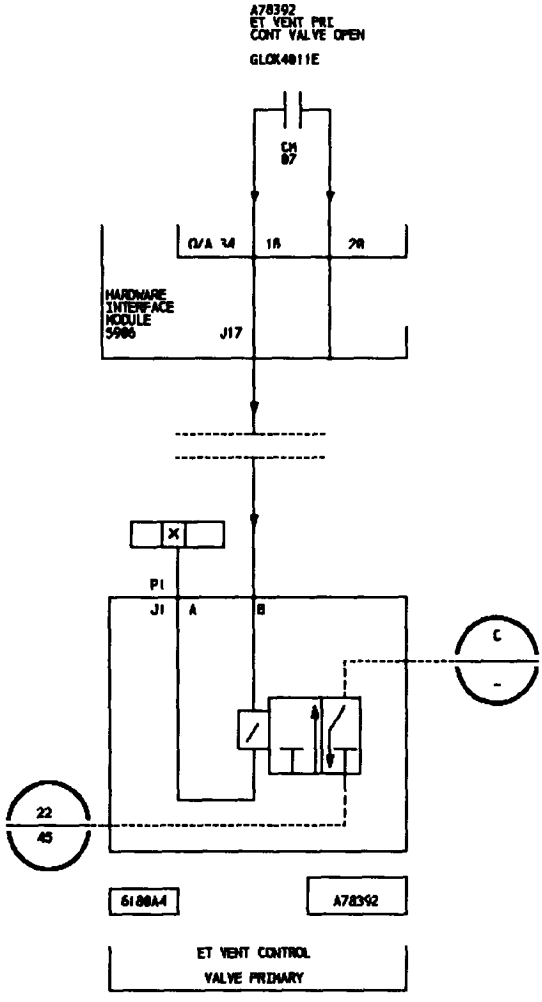


Figure 2.

**TOP-LEVEL DEFINITION**

(Defframe THING
(nomenclature "something")
(kinds SYSTEM
MEASUREMENT
COMMAND
DISCRETE OBJECT))

(Defframe DISCRETE-OBJECT
(NOMENCLATURE"a discrete object")
(also THING)
(kinds DISCRETE MEASUREMENT
DISCRETE COMMAND))

(Defframe COMMAND
NOMENCLATURE "a command")
(also THING)
(kinds DISCRETE-COMMAND
ANALOG-COMMAND))

**MID-LEVEL DEFINITION**

(Defframe DISCRETE-COMMAND
(nomenclature "a discrete command")
(also COMMAND
DISCRETE OBJECT)
(instances GLOK4011E)
(output out))

**INSTANCE-LEVEL DEFINITION**

(Defframe GLOK4011E
(nomenclature "External Tank
Control Valve Open Command")
(also DISCRETE-COMMAND)
(outputs out (CH-5986-34-07 IN)
(channel CH-5986-34-07)
(component of CARD-5986-34)
(component position 07)
(physical-location External-Tank))

(Defframe A78392
(nomenclature "ET Vent Control Valve Primary")

(also 3WAY-SOLENOID-VALVE-CLOSED
(inputs (coil (GLOK4011E- CONTACT out))
(pressure-in1 (ATMOSPHERIC-PRESSURE out))
(pressure-in2 (HE-SUPPLY-PRESSURE out)))
(outputs (pressure-out (A78394 pressure-in2)))
(physical-location ET)

Figure 3.

valve, A78392, and its associated primary open command, GLOK4011E. Instance level frame definitions for A78392 and GLOK4011E are described (Figure 3), as well as the associated top and mid level frame definitions for a command. This example highlights the fact that there is not always a one-to-one correspondence of line replaceable component-to-instance level frame definition. While Figure 2 shows only a valve and its open command, in order to completely model A78392 and GLOK4011E, the instance level file would also have to include frame definitions for GLOK4011E's associated command measurement, command contact, and Hardware Interface Module card. (Note that top level object abstractions and the mid level generic component definition for a 3-way solenoid valve are not shown.)

## KATE SHELL

### Process Monitoring

Once the knowledge base (model) accurately reflects a physical system's behavior, monitoring is relatively straightforward. Process monitoring can be broken down into two main software modules, the Measurement System and the Constraint System. During normal operation, the Measurement System continuously polls command and sensor measurements through a hardware interface. This system then places all measurements that have changed by a user-defined "significant amount" on a queue for examination by the Constraint System.

The Constraint System processes any command or sensor measurement changes. If a command has changed, the new value is propagated through the model to generate expectation values for the system sensors. The sensor measurements placed on the queue by the Measurement System are then compared to the expectation values derived from the model. If all measurements are in agreement with the expected values, the assumption is made that the system is behaving properly, and monitoring continues. If, however, any sensor measurement differs from the expectation calculated using the model, it is assumed that the discrepancy is due to a physical system failure, and the Diagnoser is invoked to localize the fault.

### Diagnosis

#### Introduction

The Diagnoser's task is to search for all possible failures within the system that can explain the current sensor readings. Reasoning from the sensor data, KATE uses a "violated expectations" approach to diagnosis which is a technique compatible with the intuitions of human diagnosticians. KATE currently assumes a single point of failure, however, the ability to diagnose to this point of failure is dependent upon the amount of sensor information available to KATE (i.e., the visibility into the system).

#### Diagnostic Algorithm

When the diagnoser is invoked due to a discrepancy, sufficient time is allotted for system stabilization. This time is given for the effects of a possible system failure to reach all of the system measurements (sensors) so that they may be used to aid in the diagnostic process. After system stabilization has occurred, any other discrepant measurements are gathered.

Initially, every component in the system which can be considered a potential "suspect" is placed on a suspect list. It is the diagnoser's task to reduce this list to a minimum number of suspects that can explain all current sensor information. These suspects are gathered using structural information found within the frames. Starting from the original discrepant measurement, all components that are structurally upstream from this sensor are considered suspect. The discrepant sensor is also placed on the list as a possibly faulty object. All other components are ignored (excluding structural faults like "bridges" or "short circuits" (2)) since objects not structurally upstream of this sensor cannot account for the discrepancy in its reading.

Once the suspects have been determined, all sensors which can be affected by these suspects are gathered. These sensors are also known as sibling sensors, and are used to aid in the effort of determining a suspect's innocence.

Further pruning of the suspect list is then attempted through the use of the functional information stored within the frames. Suspects which are of type "command" or "pseudo-object" are automatically cleared because it is assumed that these objects cannot fail. Furthermore, in the case that there exists more than one discrepant measurement, innocence can be established for the discrepant sensors based on the fact that a failed sensor cannot

cause another sensor to fail (refer to detailed description below).

A more sophisticated pruning method is then required to further localize the source of faulty system behavior. This method is based upon analysis-by-synthesis, a search for some fault that can explain what the sensors are showing, and has been labeled "The Full Consistency Algorithm" (3).

In general, the Diagnoser uses a "generate and test" paradigm. The algorithm involves calculating a hypothetical value for each suspect that would explain the discrepant measurement's value. In other words, the Diagnoser determines what state the suspect would have to be in to produce the discrepant measurement (generate). If the hypothetical state of the suspect consistent with the discrepant measurement cannot also account for the other system sensor readings (i.e., sibling sensors), then that suspect cannot be the cause of the failure and may be cleared from the suspect list (test). This hypothetical value is derived by performing symbolic inversion of the stored functional relationship between the discrepant measurement and the suspect. For further detail on the inversion process and a description of the inversion algorithm, refer to (4) and (5), respectively.

Sensor Failure and Missing Data

When the Diagnoser is invoked due to a discrepant sensor reading, KATE assembles a list of possible suspects. The discrepant sensor itself is also placed on this list as a plausible suspect. The sensor is processed just as any other suspect object in the system, and if its innocence cannot be proven, it is retained as a potential cause for its own discrepant reading.

Sensors can, in fact, be easier to diagnose than other system objects. Validation of a sensor can, in single fault environments, become a quite trivial problem. If more than one sensor is discrepant, these sensors automatically clear one another from suspicion because no one sensor can explain another sensor's discrepant reading. The fact that KATE can diagnose a sensor failure just as any other system object and, in many situations, represent an especially easy case, distinguishes KATE's model-driven approach from conventional software and traditional rule-based techniques (3).

The ability to diagnose sensors has been of particular benefit since, historically, Shuttle launches have been threatened more by instrumentation problems than by actual system failures, and unfortunately, there did not exist a fast, reliable way to distinguish the two. The original goal of the KATE project was to provide a means to determine whether a system failure indicator in fact resulted from an actual hardware failure that might be critical to continued system operation or was merely a result of a defective instrument.

Another advantage that a model-based approach offers over a conventional approach is that missing or degraded data due to a sensor failure is easily accommodated. Whereas most conventional approaches require pre-encoded actions for each combination of sensed data being present or absent, KATE does not require any special software to handle the many combinations of missing/available data. KATE simply refrains from using any sensor data that is no longer available. Sensor data is used by the Diagnoser in two ways: (1) it is compared to expectations generated by the model during normal operation to detect faulty behavior and (2) it is compared to expectations generated from fault hypotheses to confirm or disconfirm them. If the sensor data is no longer present, it is simply not compared with the expectations generated by the model. The diagnoses that follow may result in more suspects being retained than what would have been retained had the sensor information been available, but no suspect will wrongfully be cleared. In some sense, the missing sensor data is treated the same as if the sensor data had not existed in the first place (6).

Control and Redundancy Management

Conventional control techniques require that an engineer, who is familiar with the design of a specific system and is knowledgeable of the kind of devices that make up that system and how they may fail, hard-code the functionality and connectivity of the system. Programs must be written to specify which commands need to be issued to control a specific device and which measurements need to be checked to ensure the desired goal. The programmer obviously must try to foresee all possible failures that may affect control of that device and write subroutines to handle each case.

In KATE, the system's functionality and connectivity is represented in the knowledge base and is easily and rapidly accessed by the systems thereof. This allows for the capability of low level control, based on a concept somewhat similar to that used to diagnose the cause of a failure in an engineering system. The Diagnoser infers which failed component(s) would cause a discrepant condition, whereas the Control System infers the required state of all controlling components and commands that would result in a newly desired state. This conceptual relationship resulted in the natural progression from LES to the KATE project. Using this technique, KATE is capable of

accepting a high level, desired system goal from the user, consisting of an object and a value to be obtained by the object. For instance, the user may want to open or close a discrete valve, change the state of a relay, open or close an analog valve by a certain percentage, or cause a measurement or indicator to read a certain value. KATE then determines and issues the appropriate low level commands necessary to accomplish the desired task. Furthermore, redundancy is automatically activated if required by a prior component failure.

The Control System performs the task by breaking the problem or goal down into subgoals and their subgoals and so on, recursively, until controllable objects (i.e., commands) are reached. A set of options for achieving the desired state of the object is the result of this process. For further details and examples, refer to (7). The Control System dynamically creates, selects and executes the control code. Since KATE is continually monitoring the system, checks will automatically be made to ensure successful completion of the task.

In addition, the user has the option of issuing powerful, high level commands, such that certain states of the system or specific devices are "maintained" or "controlled" by KATE. In the case that a required state of an object is affected by a failure, the Control System will automatically be invoked to search for an alternate means to maintain or control back to that requirement, and issue the necessary commands to do so.

Control Advisories

The main focus of KATE is the modeling and control of tanking systems in a launch environment. In light however, of the potential of costly damage to ground and flight hardware systems and the hazards surrounding the use of cryogenic fuel in tanking operations at the Kennedy Space Center, all involved - from KATE's software system designers to launch system operators - are understandably concerned with using the KATE prototype for on-station control until the prototype has been rigorously tested and validated. Control advisories provide an alternative to autonomous control and offer a means by which to demonstrate KATE's control capability without risk to ground or flight hardware systems. Today, for demonstration purposes, control advisories are implemented as prerequisite, reactive, or conditional control logic that is initiated upon violation of system operating criteria. Under ideal conditions, initiating the Control Advisor would be as simple as making a menu selection to instruct the KATE shell to perform advisories rather than issue commands to control hardware components. Once selected, the Control Advisor would provide recommendations on control options to the operator based on operating procedures, system constraints, and knowledge of high-level goals for control of a system or subsystem. (8)

## KATE TOOLS AND UTILITIES

Generic utilities, as well as a few application specific utilities, have been made available in the KATE system. Brief descriptions of the most commonly used tools and utilities follow.

**AUTONOMOUS CAMERA CONTROL** - Provides visual confirmation of component status in response to an operator request or diagnosis of a component failure.

**CONSISTENCY CHECKER** - A knowledge base verification tool used in conjunction with the Tree Display (described below). Examines selected knowledge base slots for accuracy of information and reports any discrepancies found.

**EXPLANATION FACILITY** - Provides an explanation of the rationale used by KATE during diagnosis to indict or vindicate a component as a failure suspect.

**KATE REMOTE DATA TRANSFER UTILITY** - Allows KATE to simulate real-time data acquisition using archived data.

**PLOT** - Provides realtime line plots of component data for commands and measurements over time. Historical data plots may also be generated from archived data.

**PROCEDURE READER** - Provides the capability to enter high level goals in the form of procedural steps. KATE then executes each procedure by controlling the actual hardware or KATE's simulation of the hardware system.

**SCHEMATIC DISPLAY** - An interactive display of system schematics taken from actual engineering documents. Reports current status of any component depicted on the page.

**SIMULATOR** - Allows for simulated operation of a hardware system using a software model of the system.

**SINGLE POINT FAILURE ANALYSIS** - Analyzes the model and detects the "weak" points of a system, i.e. any component whose failure could cause loss of effective control of the system.

**TREE DISPLAY** - Provides a graphical view of the system architecture from the viewpoint of a user selected component. Displays the component and its connections to other components using the relational concepts of "upstream" (left side of the tree) and "downstream" (right side of the tree).

## CURRENT APPLICATIONS

Currently, the Artificial Intelligence Section is focusing on the following two major implementations of KATE: Autonomous Launch Operations and Shuttle Liquid Oxygen Prototype.

Autonomous Launch Operations (ALO)

KATE is being developed under the U.S. Air Force's Advanced Launch Systems (ALS) Advanced Development Program (ADP) as a project entitled Autonomous Launch Operations (ALO). The objective of ALO is to demonstrate an autonomous launch control software system that performs real-time monitoring, fault detection, diagnosis and control from high-level operations requirements. It is part of an effort to reduce overall launch operations costs, by significantly decreasing process control software development and maintenance costs and by greatly reducing launch crew size, human error and fault recovery time.

In order to demonstrate the above stated goals, two hardware models have been constructed as targets for prototyping the KATE software that is being extended in order to achieve the ALO objectives: A Water (H2O) Tanking System model and an Liquid Nitrogen (LN2) Tanking System model. In addition, the existing Environmental Control System (ECS) model hardware is expected to be integrated with the tanking systems, creating a multiple-subsystem testbed, such that all three models can be used together to demonstrate a more launch-realistic environment with KATE software systems handling a variety of launch support subsystem models.

As a first step in demonstrating the KATE software for the ALO project, the H2O Tanking System model is being used for real-time monitoring, fault detection, diagnosis and control of fluid tanking systems. As part of the H2O Tanking System demonstration objectives, KATE performs the following tanking sequences and operations through its control capabilities: Ullage pressurization, Transfer line chilldown (simulated), Main Engine chilldown (simulated), Slow fill, Fast fill, Topping, Replenish, Fill circuit drain and vehicle pressurization, Engine firing (simulated) and Drainback. During these H2O Tanking System operations/sequences KATE's ability to perform numerous tasks is demonstrated. Phase 2 demonstrations are currently being performed against this H2O Tanking System hardware. As a future step, the KATE developers will be greatly challenged by the modeling issues and the required KATE software enhancements surrounding the controlling and health monitoring of cryogenic fluid systems when they move to the LN2 Tanking System model.

Shuttle Liquid Oxygen (LOX) Prototype

During a launch countdown, launch support personnel must not only monitor the health of the system and be ready to perform troubleshooting in a tense, time-critical situation, but they must also be ever aware of system constraints, operating procedures, and operating criteria. To aid in this process KATE is currently being applied to the Shuttle Liquid OXygen (LOX) loading system. KATE-LOX is funded by the Office of Aeronautics and Space Technology. The ongoing objectives of this project are to incorporate technological advances in control, monitoring and diagnostics techniques to increase productivity and reduce operator error, as well as lower software development and maintenance cost in the shuttle ground operations environment. (9)

Since April 1990, KATE-LOX has monitored fueling operations during 18 shuttle launch countdowns in an offline mode. Although the prototype is still under development, KATE-LOX has experienced success on several occasions by accurately diagnosing failures of LOX equipment during live tankings. When completed, the KATE-LOX prototype will perform monitoring and diagnosis of the Orbiter's external tank fueling operations, and will

provide advice to operators on control options.

The LOX Expert System (LES) mentioned previously, also modeled the LOX system but was implemented using a simplified model and a simplified concept of flow. KATE-LOX employs more sophisticated control and diagnostic algorithms as well as a more complex model of the LOX hardware system that more accurately represents the effects and constraints involved in the flow of a cryogen such as liquid oxygen. Also new utilities have also been designed to reflect the needs of firing room operators.

Model validation and system testing are being accomplished by exercising the system against the Shuttle Ground Operations Simulator model of the Liquid Oxygen Loading System, online monitoring during live launch countdown loading operations and simulated loadings using the KATE Remote Data Transfer Utility.

The LOX system contains approximately 515 LOX specific data points. Two-thirds of these sensor points have been modeled, the majority or which occur at the instance level. The LOX knowledge base currently contains 1750 frames and is expected the reach 2500 by knowledge base completion. (10)

## SUMMARY AND FUTURE WORK

The development of KATE provides a means to avoid the creation and maintenance of large hard-coded programs to control and diagnose engineering system domains. Instead, knowledge bases are developed that describe the connections between internal application system components and how the outputs of each component depend upon its inputs (i.e., control relationships). Using this domain-specific model, KATE has the capability to intelligently control, monitor and diagnose faults for the particular application. The model produces control operations and expected values for the hardware system's measurements. Constraint checking is performed whenever a command or sensor value has changed. Upon detection of a discrepant sensor reading, the diagnoser is invoked. The model is additionally used to test diagnostic hypotheses generated as explanations for observed failures. Symbolic inversion of the dependency of a measurement upon each suspect component is used to calculate a hypothetical value for the suspect that could explain the discrepancy. Various consistency criteria are then used in an effort to eliminate all but one of the suspects - the culprit. This same inversion process is used for controlling objects by calculating input value(s) for an object which will result in a desired output. In addition, KATE uses both its control and diagnosis capabilities in performing redundancy management when a request to maintain a high level system goal is disrupted by a system failure. The model is automatically updated as the engineering system is manipulated or degrades. All diagnostic and control decisions are made in real-time, taking into account failed objects, objects which are being maintained, and those objects which are already at their desired states. Furthermore, modeling a system in terms of its structure and function, allows for the diagnosis of sensor failures similar to that of other system components. This approach also allows for easy accommodation of missing sensor information, and KATE can continue monitoring, diagnosis and control in the presence of a sensor failure. The result is increased machine intelligence in the area of reasoning about a system's health and controlling the state of a hardware system.

Uses for this type of system at the Kennedy Space Center include checkout of ground, payload, and launch support equipment, launch team training, and simulation of ground support and space station flight hardware systems for software component checkout.

Significant work remains to be done on improving the complex and time-consuming process of model building. Several avenues for the development of a graphical knowledge base editor and an automatic knowledge generation tool are being pursued to alleviate this problem. Should automatic knowledge generation reach fruition, a significant portion of KATE 's knowledge base could be generated automatically from Computer Aided Drawing system schematics and design notes.

Technical matters, such as modeling issues - particularly those relating to pseudo object detection and knowledge representation, improving diagnostic capability, and developing testing and validation methodologies continue to provide challenges. Increasing processing speed and preparing for integration with KSC launch processing systems are also concerns. To this end, porting to a conventional language and delivery platform is actively being pursued.

The development of KATE and its associated concepts are ongoing. With each new application of KATE, software enhancements are made to enable KATE to become more generic and encompassing in its ability to handle a wider variety of and more complex engineering systems.

## ACKNOWLEDGEMENTS

The authors would like to recognize NASA project engineer, Tim O'Brien, of the KSC Artificial Intelligence Lab, for his assistance in the preparation of this paper. We further acknowledge the Boeing Aerospace Operations, Model-Based Systems Group for their contributions in the design and development of KATE.

## REFERENCES

(1) Internal Document, "KATE Knowledge Base Generation Guide" (September 1990).

(2) R. Davis, "Diagnostic Reasoning Based on Structure and Behavior", D. G. Bobrow ed., Qualitative Reasoning about Physical Systems, MIT Press, Cambridge, MA, 1985.

(3) E. A. Scarl, J. R. Jamieson and E. New, "Model-based Reasoning for Diagnosis and Control", Proceedings of the First Florida Artificial Intelligence Research Symposium (FLAIRS-88), Orlando, FL (May 1988).

(4) E. A. Scarl, J. R. Jamieson and C. I. Delaune, "Diagnosis and Sensor Validation through Knowledge of Structure and Function", IEEE - Transactions on Systems, Man and Cybernetics SMC-17 (3), pp. 360-368 (May/June 1987).

(5) S. Thomas, "Symbolic Inversion of Control Relationships in Model-Based Expert Systems", Final Report - NASA Research Grant NAG10-0045 (December 1988).

(6) C. L. Belton-Parrish and S. Enand, "KATE - A Model-based Diagnostic and Control Shell", Intelligent Diagnostic Systems, Eds. K. F. Martin, J. H. Williams and D. T. Pham, IFS/Springer-Verlag, to be published in Fall 1992.

(7) E. New, "Knowledge-Based Control and Redundancy Management Techniques Used in NASA's KATE Project", Proceedings of Southcon/87, Orlando, FL (March 1987).

(8) T. Gould, "KATE Video Script", KATE 20 Minute Video (April 1991).

(9) C. L. Belton and B. L. Brown, "Knowledge-Based Autonomous Test Engineer (KATE) - A Model-Based Control and Diagnostic Shell", Research and Technology, 1990 Annual Report, NASA Technical Memorandum 103811.

(10) B. L. Brown, "KATE-LOX Narrative", Code RC Center Management Review (August 1991).