INTEGRATED DIAGNOSIS AND CONTROL FOR HYBRID DYNAMIC SYSTEMS

Gabor Karsai, Sherif Abdelwahed, Gautam Biswas

Institute for Software Integrated Systems

Vanderbilt University
Nashville, TN 37203, USA

{gabor,sherif,biswas}@vuse.vanderbilt.edu

ABSTRACT

In this paper we present an approach for combined fault diagnosis and reconfigurable control structure for a general class of hybrid systems. In this approach a plant is modeled using an extended version of bond graphs, namely hybrid bond graph, where discrete mode transitions are represented as binary switch junctions. A hybrid observer has been developed that uses this model to track the system behavior within and across modes. Two complementary approaches are used for fault detection and isolation. The first diagnoser is based on hybrid models and uses the hybrid observer, qualitative reasoning techniques, and real-time parameter estimation. The other diagnoser is based an abstracted discrete event model of the system that shows the causal and temporal relation between failure modes and corresponding abstract observations. To accommodate detected failure a new controller can be selected for a previously developed controller library based on the current condition. Control reconfiguration can be also achieved through online control techniques.

INTRODUCTION

Large engineering systems such as manufacturing systems, power networks, and chemical plants are usually designed for automated operation. Such automated systems are typically prone to physical (hardware) and/or logical (software) failures. In many situations, these system support critical services and failure can have serious economic, health, and security impacts. Consequently, automatic failure diagnosis forms a necessary part of these systems. Accurate and speedy diagnosis of faults is vital to the health and efficiency of the underlying system. In general, the diagnosis process aims to detect, isolate and predict possible failures by observing signals and measurements form the system sensors, comparing it with a mathematical model representing relevant nominal and/or faulty behavior, and explaining the observed behavior in terms of a hypothesis about possible abnormal changes to the state of the system components.

To ensure a high degree of reliability and safety the effects of system failures must be mitigated and control must be maintained under a variety of fault scenarios. Sophisticated control techniques are usually implemented to support the system operation under nominal conditions. If systems are designed with redundancy, control decisions have to be made about when and how backup systems should be activated, and how exactly the reconfiguration should be executed.

A large class of contemporary engineering system can be classified as hybrid systems. Hybrid systems are dynamic systems with both discrete-event and continuous-time based components. Considerable research work has been dedicated recently to the study of various aspects of hybrid systems dynamics including the issues of behavior tracking, diagnosis, and control. See for example[34] and the references therein. However, there has been very little work on integrating the diagnosis and control process in a formal way for hybrid system. Most fault-adaptive control techniques tend to take a pragmatic approach. Potential fault situations are pre-enumerated, and appropriate fault accommodation actions are built into the supervisory controller for each case. The approach works well for these cases, but may break down in unforeseen situations. Furthermore, fault-adaptive control techniques usually geared towards handling broken components. In many realistic situations, the system suffers only partial degradation and failures. If we can build online capabilities to detect and estimate these partial failures, more sophisticated control algorithms can be designed to keep the system operational under these conditions[31,32,33].

For the DARPA SEC project, we are developing a systematic model-based approach to the design and implementation of control systems that can accommodate faults. We call this approach Fault-Adaptive Control Technology (FACT). Developing fault-adaptive control requires us to solve a number of technical problems beyond the capabilities of traditional control approaches. First, faults must be detected while the system is in operation. System dynamics is complex, and sensors can be noisy, therefore, differentiating degraded faulty behavior from nominal behavior of the plant quickly is a non-trivial problem. Fault detection must be followed by rapid fault isolation and estimation of the fault magnitude. Then a decision has to be made online on how to reconfigure the control system to accommodate the fault. Many alternatives may have to be evaluated, and metrics will have to be defined that either (1) select an optimal configuration, if it can be computed in a feasible manner, or (2) the best possible reconfiguration is derived under given time and resource constraints. Finally, the reconfiguration must be executed, which means that set points and control parameters may have to be changed, or a different controller may have to be selected to continue system operation. The challenge is bring together methodologies from fault diagnostics, control theory, signal processing, software engineering and systems engineering to build the integrated online FACT system.

In this paper, we present the developed fault adaptive control structure and describe the different techniques implemented for failure diagnosis and control reconfiguration. In Section 2, we present a reference-architecture for FACT systems. Section 3 presents the different models that are used to describe aspect of the system behavior and functionalities that are relevant to the fault adaptive control architecture. Section 4 describes the hybrid observer scheme for tracking nominal system behavior. Section 5 discusses the fault isolation methodologies. Preliminary results that demonstrate the effectiveness of our approach are presented. Section 6 briefly discusses fault-adaptive control and controller reconfiguration. The summary and conclusions appear in Section 7 of the paper. We illustrate the basic modeling concepts and our diagnosis algorithms using a two-tank system as the plant, with a supervisory controller.

FAULT ADAPTIVE CONTROL ARCHITECTURE

The fault adaptive control architecture, shown in Fig. 1, is an integrated structure of model-based and logical approaches for fault detection and isolation, parameter estimation, and control reconfiguration for a general class of hybrid systems. In this architecture, the plant is observed and managed through a set of monitoring and reconfigurable control modules. Hybrid models[2], derived from hybrid bond graphs[3] systematically integrate continuous and discrete system dynamics and discrete events to establish the core of the modeling framework. Supervisory controllers, modeled as an extended finite state machine, are used to generate the discrete events that cause reconfigurations in the continuous energy-based bond graph models of the plant. Fault detection involves comparison of the expected behavior of the system generated from the hybrid models with actual system behavior, to determine when discrepancies occur. This requires the design and implementation of hybrid observers that estimate the continuous dynamic states of the system and detect mode transitions in the system operation. Sophisticated signal analysis and filtering methods linked to the hybrid observers are used for detecting deviations from nominal behavior and triggering the fault isolation schemes.

Our diagnostic schemes integrate the use of failure-propagation graph based techniques for discrete-event diagnosis[4] and combined qualitative reasoning and quantitative parameter estimation methods for parameterized fault isolation[5] of degraded components (sensors, actuators, and plant components). The dynamic system state accumulated from the observer (discrete system mode plus continuous state vector) and fault isolation units (status of faulty and degraded sensors, actuators, and plant components) define the active system state model. The tracking, fault detection, and fault isolation mechanisms, shown in Fig. 1, together constitute a bottom-up computational approach for estimating the dynamic system state (nominal or faulty) by monitoring plant and controller variables.

The reconfiguration controller uses this information to select from the controller library the controller that is most effective in maintaining desired system operation and performance. This requires the definition of metrics and decision criteria that govern the controller selection process. The selection and reconfiguration mechanisms operate in a top-down manner, using the dynamic state information to effect changes in supervisory control mechanisms, such as selection (not synthesis) of feedback control mechanisms, and re-tuning of low level regulators, such as PID or model-based controllers. The overall computational architecture combines the bottom-up and top-down computational schemes in a seamless manner, via the shared active model.

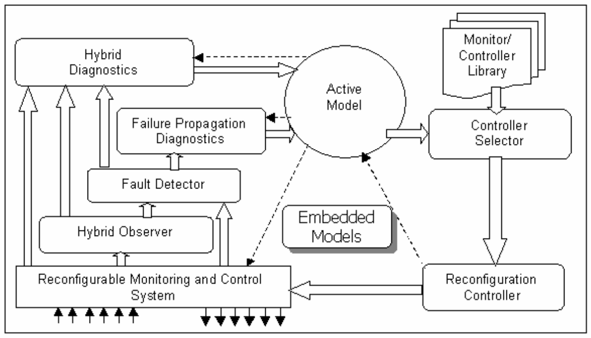
Hybrid Diagnostics
Active Model
Monitor/ Controller Library
Failure Propagation Diagnostics
Controller Selector
Fault Detector
Embedded Models
Hybrid Observer
Reconfigurable Monitoring and Control System
Reconfiguration Controller

Figure 1: Fault Adaptive Control Architecture

Alternatively, online control and supervision can be implemented to ensure a given safety specification under both nominal and faulty conditions. The safety control problem requires the system to move to a predetermined safe region from a given set of initial states in the state space of the system. The online approach does not require the existence of a finite quotient equivalent for the system and therefore is applicable to complex hybrid systems. Moreover, the approach can be adapted to accommodate possible changes in the system parameters that may occur as a result of a fault or parameter changes in time-varying systems.

The implementation and support for the online FACT architecture is based on our model-integrated computing paradigm[1]. To achieve this, we have created (1) a graphical modeling environment that facilitates building hybrid models of the plant and controllers, and (2) a set of run-time components that can execute the code synthesized from the models. This code, when integrated with the generic FACT run-time components, instantiate the architecture for a specific application domain.

THE MODELING PARADIGMS

In general, different aspect of the systems behavior and functionalities can be used for failure analysis and control reconfiguration. The FACT architecture supports two basic modeling paradigms that can capture essential system dynamics from different but related prospective. The detailed mixed continuous and discrete behavior in the plant components is captured through Hybrid bond graphs. Fault propagation graph, on the other hand, focuses on the causal and temporal relationship between different operation regions (typically corresponds to failure modes) and the associated abstract observations. In addition to these two modeling structure, we use extended state machine to model the high level supervisor.

Hybrid Bond Graphs

In the FACT architecture plant components are modeled as bond graphs. Bond graphs represent energy-based models of the system in terms of the effort and flow variables of the system. Bonds specify interconnections between elements that exchange energy, which is given by the rate of flow of energy, *power = effort x flow*. Bond graphs represent a generic modeling language that can be applied to a multitude of physical systems, such as electrical, mechanical, and thermal systems. There exist standard techniques to build bond graph models of systems based on physical principles. State equations can be systematically derived from the bond graph representation of the system. Temporal causal graphs, the models for qualitative diagnostic analysis, can be systematically produced from bond graphs[15]. An extended version of bond graphs, referred to as hybrid bond graphs (HBG)[3] is used to model possible discrete transitions in system behavior

Timed Failure Propagation Graphs

Timed failure propagation graphs (TFPG)[4] are causal models that describe the system behavior in presence of faults. The timed failure propagation graph is a labeled directed graph where the nodes represent either failure modes - which are fault causes - or discrepancies - which are off-nominal conditions that are the effects of failure modes. Discrepancies can either be monitored (attached to alarms) or silent, and depending on the way it is triggered by the incoming signals it is further classified as either ``AND" or ``OR" discrepancy. Attributed edges between nodes in the graph represent causality, and the attributes specify the temporality of causation given by an upper and lower time constraints on the propagation of failure between nodes.

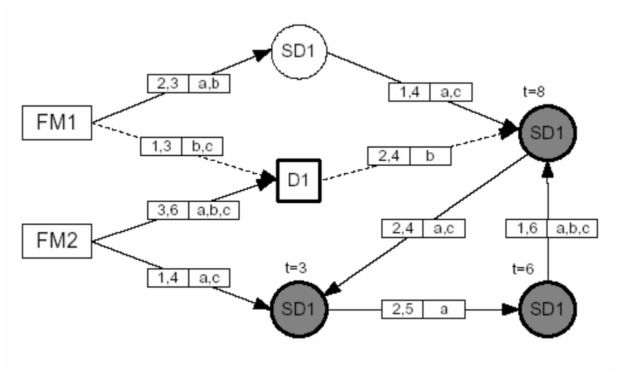

Figure 2: A hybrid failure propagation graph

An extended version of TFPG model, referred to as hybrid failure propagation graph, is implemented. The hybrid failure propagation graph allows the representation of failure propagation in multi-mode (switching) systems in which the failure propagation depends on the current mode of the system. To this ends, edges in the graph model can be constrained to a subset of the set of possible operation modes of the system. Formally, a hybrid failure propagation graph model is represented as a tuple $G = (F, D, E, M, ET, EM, DC, DS), where F is a nonempty set of failure nodes, D is a nonempty set of discrepancy nodes, with $F \cap D = \emptyset$, $E \subseteq V \times V$ is a set of edges connecting the set of all nodes $V = F \cup D$, M is a nonempty set of system modes, $ET: E \rightarrow I$ is a map that associate every edge in E with a time interval, $EM: E \rightarrow M$ is a map that associate every edge in E with a set of modes in M, $DC: D \rightarrow$ {AND,OR} is a map defining the class of each discrepancy as either AND or an OR node, $DS : D \rightarrow$ {ON, OFF} is a map defining the monitoring status of the discrepancy. An example of a hybrid failure propagation graph is shown in the above figure.

Supervisory Controllers Models

In the FACT architecture, the reconfigurable monitoring and control component represents all the traditional monitoring and control functions in an application. We envision that this component is implemented mainly in software, although some components might utilize dedicated hardware components. This component is also "reconfigurable": its sub-components, their parameters, and their interconnection can be changed during system operation.

To represent this reconfigurable monitoring and control component, we have developed a modeling language, called Controller Modeling Language (CML). The approach followed here is that of Model-Integrated Computing[1]. CML represents controllers on two levels; the regulatory level, and the supervision level. On the *regulatory* level, it represents controllers using computational blocks that form a signal flow diagram. The signal flow diagram has process-network semantics: each block is a process that is scheduled for execution upon arrival of data on its inputs. Then the process performs some calculations and may generate output data that is sent to downstream blocks. After finishing processing, the process terminates and waits for the next triggering data. On the *supervisory* level, it represents controllers using an extended finite state machine model similar to that of Statecharts[22].

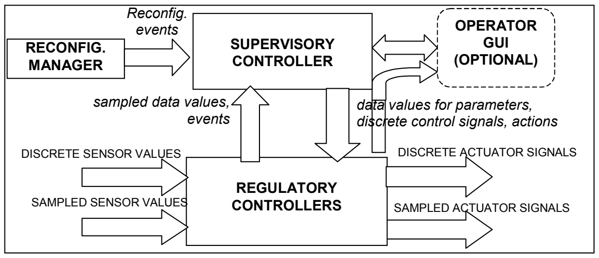

Figure 3: Relationship between the supervisory and regulatory controllers

The relationship between the two controller layers: supervisory and regulatory, is shown in Fig. 3. The regulatory layer operates in a discrete-time fashion, i.e., it receives discrete (sporadic) and sampled data from the plant, and it generates discrete (sporadic) and sampled data for the actuators. On the other hand, the supervisory controller operates in a discrete-event mode, i.e., it has no explicit notion of time. It receives sampled data values and discrete events generated in the regulatory layer, and sends new data values for parameters, and events in the form of discrete control signals to the regulatory layer. The supervisory controller can also trigger the execution of reconfiguration actions. As mentioned above, during reconfiguration the design procedures associated with the regulatory blocks will be triggered to recalculate parameter values.

THE HYBRID OBSERVER

The hybrid observer tracks the system behavior across different modes of operation. This involves two steps: Tracking continuous system behavior in individual modes of operation, and Identifying and executing all mode changes including controlled and autonomous jumps. Transitioning from one mode to the other involves: (i) switching the state equation model that defines continuous behavior in a mode, and (ii) applying the reset function to derive the initial state in the new mode.

The observer uses the state equations models - derived by symbolic analysis from the hybrid bond graph model - for tracking the continuous behavior in a particular mode of operation. The analysis also derives the controlled and autonomous events that define mode transition conditions as the system behavior evolves in time. Solving for the mode transitions requires access to controller signals for controlled jumps, and predictions of state variable values for autonomous jumps. We rewrite all autonomous jump conditions in terms of the state variables of the system. The state variable estimates are obtained from the hybrid observer, and these values are used to determine if autonomous jumps have occurred. If a mode change occurs in the system, the observer switches the tracking model (to a different set of state space equations), initializes the state variables in the new mode (using a "reset" function, again derived from the hybrid bond graph model), and continues to track system behavior with the new model[13]. Since the input and output of the system may be affected by processor disturbances and measurement noise, we use a Kalman filter[23] to track system behavior in a single mode of operation. For a given state space model the Kalman gain matrix can be computed from the covariance matrices, as usual.

FAULT DETECTION AND ISOLATION

A primary component of our system is the model-based fault detection and isolation (FDI) subsystem that can deal with sensor, actuator, and parametric faults in the system. Traditional FDI methods[6,24,25,26] are primarily directed toward additive faults that include failures in sensors and actuators. Isolation of parametric component faults, which are multiplicative, requires the use of sophisticated parameter estimation techniques[26]. Numerical techniques for state and parameter estimation often face convergence and accuracy problems when dealing with high-order models that may contain non-linearities[7,26]. Parameter estimation techniques are often biased by measurement noise, and may need specialized approaches to compensate for these situations[26,29]. Accurate parameter estimation also requires persistent excitation of the input, and this may not always be true during system operation. Furthermore, these schemes are applicable in continuous real-valued spaces, and they do not easily extend to situations where mode transitions cause discontinuous changes in the system models and system variables. Discrete-event based diagnosis techniques have been proposed, but they require the pre-compiling of the fault models and fault trajectories into Finite State Machines (FSM-s) for tracking nominal and faulty system behavior[8,9]. In the section below we will show how an alternative representation form can be used which does not require the explicit construction of FSMs.

When one deals with hybrid systems that include discrete transitions, extending these continuous methodologies becomes intractable, because the residual transformation functions have to be pre-computed for all modes of operation. Further, when faults occur, predicting the true system mode in itself becomes a challenging task. The fault isolation problem becomes even more complex, when the fault occurs in an earlier mode, but is detected in a later mode of operation. The predicted mode sequence may no longer be the true mode sequence the system goes through after the occurrence of the fault. Additional methods have to be introduced for detecting mode transitions, switching the system model when such transitions occur, and correctly initializing the system state, so that the fault observers perform correctly. Typically mode changes introduce discrete effects that cause transients, and it may be difficult to separate the fault transients from the transients caused by mode changes. Therefore, extending continuous FDI schemes to hybrid systems is a non-trivial task.

We use two approaches to the FDI problem that generalize traditional approaches: (i) the use of a robust qualitative fault isolation scheme based on tracking fault transients combined with a parameter estimation scheme for refining fault hypotheses, and (ii) fault diagnostics based on discrete event models represented as fault propagation graphs. We discuss each of these methodologies in greater detail next.

Diagnosis using Hybrid Models

Our diagnosis methodology consists of three mains steps, (i) using a hybrid observer to track system behavior, (ii) detecting fault occurrences, and (iii) isolating faults in the system. The hybrid observer, discussed in the last section, uses the models of the system to track system behavior. The fault detection schemes that compare the measurements made on the system and the predictions from the observer to look for significant deviations in the observed signals are discussed elsewhere[14]. Our fault detectors for continuous systems have to be modified to signal faults only when abrupt changes cannot be attributed to mode changes[11,13].

The overall scheme for hybrid diagnosis is illustrated in Fig. 6. We overcome limitations of quantitative schemes by combining robust qualitative reasoning mechanisms with quantitative parameter estimation schemes for parametric fault isolation[5]. Hybrid bond graphs models discussed in Section 3 form the basis for generating parameterized Timed Causal Graphs (TCG-s), a representation that captures system dynamics as causal links between system variables, annotated by temporal relations, such as instantaneous effects and integral relationships[9]. The bond graph representation explicitly includes component parameters that govern system dynamics as resistive, capacitive, inertial, transformation, and signal propagation elements. The TCG representation makes explicit the effect of changes in parameter values on the dynamics of system variables. The fault isolation methodology for hybrid systems is broken down into three steps. It includes

1. A fast *roll back process* using qualitative reasoning techniques to generate possible fault hypotheses. Since the fault could have occurred in a mode earlier than the current mode, fault hypotheses need to be characterized as a two-tuple <*mode, fault parameter*>, where mode indicates the mode in which the fault occurs, and fault parameter is parameter of the implicated component whose deviation possibly explains the observed discrepancies in behavior.
2. A quick *roll forward process* using progressive monitoring techniques to refine the possible fault candidates. The goal is to retain only those candidates whose fault signatures are consistent with the current sequence of measurements. After the occurrence of a fault, the observer's predictions of autonomous mode transitions may no longer be correct, therefore, determining the consistency of fault hypotheses also requires the fault isolation unit to roll forward to the correct current mode of system operation.
3. A *real-time parameter estimation process* using quantitative parameter estimation schemes. The qualitative reasoning schemes are inherently imprecise. As a result a number of fault hypotheses may still be active after Step 2. We employ a least squares estimation technique on the input-output form of the system model to estimate consistent values of the fault parameter that is consistent with the sequence of measurements made on the system.

Diagnosing using Timed Failure Propagation Graphs

The diagnostic system operates on the TFPG model and characterizes the fault status (actual current state) of the system by hypothesizing about the faults in components and sensors based on the signals received from the sensors and the current mode of the system. The diagnoser uses the TFPG model and the timed sensor/mode-switching signals to generate a set of logically valid hypotheses of the current state of the system. The hypotheses are then ranked according to certain criterion based on the number of supporting alarms versus the number of inconsistent one. The set of hypotheses with the highest rank are selected as the most plausible estimations of the current state of the system.

The diagnoser is implemented as a reactive module that is triggered by signals from the set of active sensors and well as mode-switching signals. A diagnoser input signal is represented by an event structure (e, t), where e denoted a monitored alarm being activated or a mode-switching signal and t is the time at witch the signal is observed. The event structure (e,t) is triggered whenever the state of a discrepancy is changed or the system changes mode. he diagnoser responds to input signals by generating hypothesis. Each hypothesis is an evaluation of the status of a failure mode. The hypothesis structure contains information about the corresponding failure mode, the estimated time of failure, and the set of supporting and inconsistent alarms as conceived based on the failure graph structure. In addition to generating and updating hypotheses, the diagnoser also generates a list of false alarms, namely those alarms that could not be explained by any hypothesis based on the timing and structure of the failure propagation graph. Figure 4 shows a simplified UML diagram of the basic elements of the TFPG diagnosis system and the relation between them.

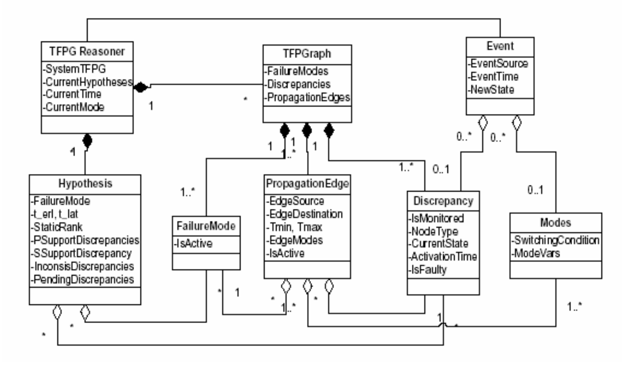

Figure 4: Core classes and their operations in the TFPG diagnosis system

In reasoning about the faults the diagnoser uses the principles of parsimony. In general, due to possible structural redundancy in the TFPG model, there can be several explanations of a give sequence of sensor signals. The principle of parsimony suggests that the simplest explanation is the best. By simplest we mean the one that involve the least number of faulty components. In general, there may not be a unique simplest explanation. In this situation the diagnoser will provide all the simplest explanations to the user. At the occurrence of every event, the diagnoser updates the set of hypothesis and the faulty components will be identified. The diagnoser updates the set of possible hypotheses about the system state based on the causal and timing consistency between the discrepancies.

CONTROLLER IMPLEMENTATION

Our approach is to develop a library of controllers, which is indexed by sets of characteristics. The goal is to use the information about current system state, i.e., the current mode of operation and system state vector along with failed and degraded states of components and subsystems to select a controller that best suits current and long term performance objectives.

We address the controller reconfiguration task on two levels. At the supervisory (discrete) level, reconfiguration implies modification of high-level control actions. This can take the form of replacing a current action sequence by a new sequence, or altering the sequence of actions in the current set. This type of reconfiguration requires that the supervisory control logic be explicitly represented as a data structure. Our challenge is to adopt model-based approaches to representing supervisory control programs, and to develop reconfiguration procedures governed by different kinds of fault conditions. At the lower (continuous) level of control, the system relies on regulators, which can range from simple switching controllers, to PID mechanisms, and then model-based controllers. Reconfiguration at this level can take on three different forms.

1. Set point changes for handling simple fault situations, such as a partially degraded component.
2. Controller tuning for handling cases where the fault changes the plant dynamics (e.g., changes in the capacitive and inertial parameters in the plant), and the re-tuning of the controller is a viable solution.
3. Structural changes (i.e., rewiring or replacing the regulators) may compensate for complex faults where the current controller architecture is unable to maintain the desired control because of a significant fault (e.g., sensor faults, actuator faults, and major structural changes in the plant, such as pump failures or valves stuck at closed).

There is an interesting and highly relevant aspect of controller reconfiguration that is also being addressed: the explicit management of reconfiguration transients. Early results[24,25] show that there are a number of techniques available for mitigating reconfiguration transients in control systems. If the selected approach of controller re-initialization and/or blending does not meet the requirements for the reconfiguration dynamics other, more explicit transient suppression techniques can be applied to mitigate the effects of switching.

Online Safety Control

An online approach to the safety control of hybrid systems has been developed in[34]. The safety control problem requires the system to move to a predetermined safe region from a given set of initial states in the state space of the system. The proposed approach can be applied efficiently for hybrid systems with small number of switching inputs. Moreover, the approach is robust to limited domain changes in the system parameters that may occur as a result of a fault or parameter changes in time-varying systems. The proposed online supervision algorithm explores only a limited part of the system state space and selects the next input based on the available information about the current state. For the safety control problem, the selection of the next step is based on a given distance map that defines how close the current state is to the safe region.

The online supervision algorithm starts by constructing the tree of all possible future states from the current state up to a specified depth. To avoid the Zeno effect, in which the controller may try to preempt time indefinitely through continuous switching, we require that any input switching event is followed by at least one sampling period. The exploration procedure identifies the set of states with the minimal distance from the safe region based on the given distance map. A state is then chosen from this set based on certain optimality criterion (for example, minimal input switching), or simply picked at random. The chosen state is then traced back to the current state and the event leading to the least distance is used for the next step. Conditions for the online controllability of system with respect to the safety specifications is established and used to provide an upper limit for the accuracy error of the online controller.

CONCLUSIONS AND FUTURE WORK

We have applied our continuous and discrete FDI methodology to diagnosing faults in a two-tank system with a number of valves. A simple supervisory controller model took the system through a number of filling, emptying, and mixing cycles. We were successful in tracking continuous system behavior through discrete mode changes, and isolating faults when they occurred, with the discrete and continuous diagnostics algorithms. As a next step, we would like to extend the two diagnostic algorithms to work in a more cohesive fashion, and inform each other as they come up with fault hypotheses. Once this step is completed, we will introduce the controller selection mechanisms to have a comprehensive implementation of the FACT architecture that has been presented in this paper.

We are also looking at applying this technology to more real-world applications, such as the fuel transfer system in modern aircraft. The physical components of the fuel system include a number of tanks, interconnecting pipes, valves, and pumps. In addition, the system is equipped with sophisticated controls to support reliable and robust fuel delivery under a variety of flight conditions, at the same time ensuring that the gravity of aircraft center of gravity is not compromised. In addition, the controllers have to deal with a number of fault scenarios, such as pump failure and pipe leaks. The goal under such conditions is not to compromise aircraft safety, but to save as much fuel as one can to continue the current mission. To achieve this, the system should utilize built-in redundancy mechanisms to compensate for the failure, and at the same time, maintain control. We are currently developing models of a generic aircraft fuel system, and testing and validating the FACT tools and techniques on a number of example scenarios that have been generated using a high fidelity simulator.

ACKNOWLEDGEMENTS

The DARPA/ITO SEC program (F33615-99-C-3611), and The Boeing Company have supported the activities described in this paper. We would like to thank Dr Kirby Keller and Mr. Mark Kay for their help.

REFERENCES

[1] Sztipanovits, J., Karsai, G.: “Model-Integrated Computing”, *IEEE Computer*, pp. 110-112, April, 1997.

[2] Branicky, M.S., V. Borkar, S. Mitter, 1994. “A Unified Framework for Hybrid Control: Background, Model, and Theory,” *Proceedings of the 33rd IEEE Conference on Decision and Control*, Lake Buena Vista, FL, Paper No. LIDS-P-2239.

[3] Mosterman P.J. and G. Biswas, 1998. “A theory of discontinuities in physical system models,” *Journal of the Franklin Institute*:335B, pp. 401-439.

[4] Misra A., Sztipanovits J., and Carnes J., 1994. “Robust Diagnostics: Structural Redundancy Approach,” Knowledge Based Artificial Intelligence Systems in Aerospace and Industry, SPIE's Symposium on Intelligent Systems, Orlando.

[5] Manders E.J., S. Narasimhan, G. Biswas, and P.J. Mosterman, 2000. A combined qualitative/quantitative approach for efficient fault isolation in complex dynamic systems. *4th Symposium on Fault Detection, Supervision and Safety Processes*, pp. 512-517.

[6] Patton, R.J., Frank, P.M., and Clark, R.N. (eds.), 2000. Issues of Fault Diagnosis for Dynamic Systems, Springer-Verlag, London, U.K.

[7] Chen, J. and Patton, R.J. 1999. *Robust Model-Based fault Diagnosis for Dynamic Systems,* Kluwer Academic, Boston, MA.

[8] Sampath, M. et al., 1996. “Fault Diagnosis using Discrete-Event Models,” *IEEE Trans. On Control Systems Technology*: 4(2), pp. 105-124.

[9] Lunze, J. 1999. “A Timed Discrete Event Abstraction of Continuous Dynamic Systems,” *Intl. Journal of Control*: 72, pp. 1147-1164.

[10] Alur, R. et al., 1993. Hybrid Automata: an algorithmic approach to the specification and verification of hybrid systems, in, R.L. Grossman, et al., eds., Lecture Notes in Computer Science, Springer, Berlin, 736, pp. 209-229.

[11] Narasimhan, S. and Biswas, G. 2000. Using Supervisory Controller Models for more Efficient Diagnosis of Hybrid Systems. Submitted to *Hybrid Systems: Control and Computation, Intl. Workshop,* Rome, Italy.

[12] Rosenberg, R.C. and Karnopp, D.C. 1983. *Introduction to Physical System Dynamics*, McGraw Hill, NY.

[13] Narasimhan, S., Biswas, G., Karsai, G., Pasternak, T., and Zhao, F., 2000. “Building Observers to Handle Fault Isolation and Control Problems in Hybrid Systems,” *Proc. 2000 IEEE Intl. Conference on Systems, Man, and Cybernetics*, Nashville, TN, pp. 2393-2398.

[14] Manders E.J., Mosterman, P.J., and Biswas, G., 1999. Signal to symbol transformation techniques for robust diagnosis in TRANSCEND, *Tenth International Workshop on Principles of Diagnosis,* Loch Awe, Scotland, pp. 155-165.

[15] Mosterman P.J. and Biswas G., 1999. Diagnosis of Continuous Valued Systems in Transient Operating Regions, *IEEE Trans. on Systems, Man and Cybernetics*:29, pp. 554-565.

[16] Pasternak, T. *Extended Relational Models for Diagnosis*, Masters Thesis, Vanderbilt University, August 2000.

[17] Lunze, J., *Diagnosis of Quantized Systems by Means of Timed Discrete-Event Representation*, in Proc. Of Thirds International Workshop on Hybrid Systems, Computation and Control, Lecture Notes in Computer Science, volume 1790, pages 258-271, March 2000.

[18] Simon, G., Kovácsházy, T., and Péceli, G., 2000. “Transients in Reconfigurable Control Loops,” *IEEE Instrumentation and Measurement Technology Conference, IMTC/2000*, Baltimore, Maryland, USA.

[19] Simon, G., Kovácsházy, T., and Péceli, G., 2000. “Transient Management in Reconfigurable Systems,” *International Workshop on Self Adaptive Software*, Oxford University, England.

[20] Pierce, C. S. "Note B: The Logic of Relatives." In Studies in Logic by Members of the Johns Hopkins University Boston: Little Brown and Co. 1883

[21] Ledeczi A., Bakay A., Maroti M.: Model-Integrated Embedded Systems, in Robertson, Shrobe, Laddaga (eds) Self Adaptive Software, Springer-Verlag Lecture Notes in CS, #1936, February, 2001.

[22] David Harel, Michal Politi: Modeling Reactive Systems with Statecharts: The Statemate Approach, McGraw-Hill, 1998.

[23] A. Gelb, Applied Optimal Estimation, MIT Press, Cambridge, MA, 1979.

[24] H.L. Jones, Fault Detection in Linear Systems, Ph.D. thesis, Massachusetts Inst. of Technology, 1973.

[25] R. Mangoubi, Robust Estimation and failure Detection, A Concise treatment, Springer Verlag, New York, NY, 1998.

[26] J.J. Gertler, Fault Detection and Diagnosis in Engineering Systems, Marcel Dekker, Inc., 1998.

[27] S. Narasimhan, P.J. Mosterman, and G. Biswas, “A Systematic Analysis of Measurement Selection Algorithms for Fault Isolation in Dynamic Systems,” 9th Intl. Workshop on Principles of Diagnosis, Cape Cod, MA, pp. 94-101, May 1998.

[28] P.J. Mosterman and G. Biswas, “Towards procedures for systematically deriving hybrid models of complex systems,” *Hybrid Systems: Computation and Control, Third Intl. Workshop*, Lecture Notes in Computer Science, vol. 1790, N. Lynch and B. Krogh, eds., Springer Verlag, Berlin, Germany, pp. 324-337, March 2000.

[29] L. Ljung, *System Identification: Theory for the user*, Prentics Hall, Englewood Cliffs, NJ, 1987.

[30] X.D. Koutsoukos, P.J. Antsaklis, J.A. Stiver, and M.D. Lemmon, “Supervisory Control of Hybrid Systems,” *Proceedings of the IEEE: Special Issue on Hybrid Systems*, P.J. Antsaklis, ed., pp. 1026-1049, 2000.

[31] P. J. Antsaklis and K. M. Passino, "Introduction to Intelligent Control Systems with High Degree of Autonomy,” Introduction to Intelligent and Autonomous Control, P.J.Antsaklis and K.M.Passino, Eds., Chapter 1, pp. 1-26, Kluwer Academic Publishers, 1993.

[32] P. J. Antsaklis, "Defining Intelligent Control,” Report of the Task Force on Intelligent Control, P.J Antsaklis, Chair. In IEEE Control Systems Magazine, pp. 4-5 & 58-66, June 1994.

[33] P.J.Antsaklis, K.M.Passino and S.J.Wang, "An Introduction to Autonomous Control Systems,” *IEEE Control Systems*, Vol 11, No 4, pp 5-13, June 1991.

[34] P. J. Antsaklis, editor. Special Issue on Hybrid Systems. Proceedings of the IEEE. 2000.

[35] S. Abdelwahed, G. Karsai, and G. Biswas, “Online Safety Control of a Class of Hybrid Systems.” 41st IEEE Conference on Decision and Control, Las Vegas, NV, 2002