IMTC 2003 – Instrumentation and Measurement
Technology Conference
Vail, CO, USA, 20-22 May 2003

Model-Based Fault-Adaptive Control of Complex Dynamic Systems

Gyula Simon[1,2], Gábor Karsai[1], Gautam Biswas[1], Sherif Abdelwahed[1], Nagabhushan Mahadevan[1], Tivadar Szemethy[1], Gábor Péceli[2] and Tamás Kovácsházy[2]

[1]Institute for Software Integrated Systems
Vanderbilt University, PO Box 1829 Station B, Nashville, TN 37235,USA
{simon,gabor,biswas,sherif,nag,tiv}@isis-server.isis.vanderbilt.edu

Department of Measurement and Information Systems,
[2]Budapest University of Technology and Economics, H-1521 Budapest, Hungary
{peceli,khazy}@mit.bme.hu

Abstract *– Complex control applications require capabilities for accommodating faults in the controlled plant. Fault accommodation involves the detection and isolation of faults, and then taking appropriate control actions to mitigate the fault effects and maintain control. This requires the integration of fault diagnostics with control in a feedback loop. This paper discusses a generic framework for building fault-adaptive control systems using a model-based approach, with special emphasis on the modeling schemes that describe different aspects of the system at different levels of abstraction and granularity. The concepts are illustrated by a fault adaptive notional fuel system control example.*

I. INTRODUCTION

Today's complex systems, like high-performance aircraft require sophisticated control techniques to support all aspects of operation: from flight controls through mission management to environmental controls, just to give a few examples. Any real-life system is prone to failures: either physical (hardware) or logical (software). When a high degree of reliability and safety is desired the effects of these failures must be mitigated and control must be maintained under all fault scenarios. In order to manage fault scenarios, we need to make a series of decisions and control actions: (1) the fault has to be detected, (2) the fault source has to be identified and the magnitude of failure estimated (e.g., a partial degradation versus a total failure), (3) depending on the nature of failure, a new control algorithm has to be selected that compensates for the partial or complete failure, (4) the plant has to be reconfigured, and (5) the new control algorithm has to be chosen. All these decisions must be made by a control system that incorporates not only simple regulatory loops and the supervisory control logic, but also a set of components that detect, isolate, and manage faults, in coordination with the control functions.

In this paper, a systematic model-based approach is presented that can create control systems capable of accommodating faults. We call this approach Fault-Adaptive Control Technology (FACT). Developing fault-adaptive control gives rise to a number of technical challenges that go beyond the capabilities of traditional control approaches. Faults must be detected while the system is in operation, followed by quick fault isolation and estimation of the fault magnitude. The next step is on line decision-making 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 select the optimal or the best reconfiguration. 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 deployed to continue system operation. The challenge is to build an integrated online approach that combines fault diagnostics, control theory, signal processing, software engineering, and systems engineering. The different components of the system require different kinds of models with different granularity and abstraction levels. Some of these models are specified by the system designer, while the computational models for on-line analysis can be automatically generated using model transformation tools.

This paper introduces the modeling paradigms, and the main building blocks of a model-based FACT architecture for complex dynamic systems. Section II presents FACT architecture. In Section III the modeling paradigms and the different models used are discussed. Section IV illustrates the operation of the system through an example: a fault-adaptive fuel control system.

II. FACT ARCHITECTURE

The overall FACT approach, illustrated in Fig. 1, is centered on model-based approaches for fault detection, fault isolation and estimation, and controller selection for hybrid systems. The plant is assumed to be a hybrid system, i.e. it combines continuous dynamic behavior in individual modes with discrete modes changes [1]. The mode changes can be *autonomous* (e.g.: water level in a tank reaching the level of a drain pipe), or they can be attributed to control commands (*con-*

0-7803-7705-2/03/$17.00 ©2003 IEEE

trolled mode change, e.g.: open or close a valve). The *Reconfigurable Control Unit* for the plant control operates in two layers: (i) the regulatory layer interfaces with the physical plant through sensors and actuators, and (ii) the supervisory layer, which is responsible for the high-level control strategy.

The heart of the *Fault Adaptive Control Unit* is the *Hybrid Observer* that tracks the behavior of the plant under nominal conditions. When the *Fault Detector* detects a discrepancy between the measured and the expected behavior, the diagnosis units are triggered. The *Discrete Diagnosis* and the *Hybrid Diagnosis* units use qualitative reasoning approaches, but they are based on models of different accuracy and resolution. The *Discrete Diagnosis* unit uses Temporal Fault Propagation Graphs describing the effects of faults to the system at a high level, while the *Hybrid Diagnosis* unit uses detailed models of the plant in the form of the Temporal Causal Graphs that captures the transient dynamics after fault occurrence. The possible fault candidates provided by the diagnosis units are merged and ranked in the *Fusion Unit*. The best candidates are then passed to the quantitative *Parameter Estimation Unit* that reduces the candidate set to a single fault candidate by computing the degree of degradation, and retaining that candidate that has the least prediction error. The results of the fault diagnosis are used to select the optimal controller. In the current system, we assume a library of controllers indexed by sets of characteristics is available. The controllers are parameterized by the current mode of operation, the system state vector, and the failed and degraded states of components and subsystems. The selection function is set up so that the new controller best meets the current and long-term performance objectives.

The *Reconfigurable Controller*'s task is addressed at two levels. At the supervisory (discrete) level, reconfiguration implies modification of high-level control actions. At the lower (continuous) level of control, the system relies on regulators. Reconfiguration at this level can take on three different forms: (i) set point changes, (ii) controller tuning, and (iii) structural changes. The *Reconfiguration Manager* is responsible for identifying the necessary reconfiguration tasks and initiating the reconfiguration process. Since the reconfiguration may lead to the introduction of large switching transients into the system, the *Transient Manager* performs the actual reconfiguration in a way that undesired transient effects are reduced [2]. Based on the result of the parameter estimation the *Plant Model* is also modified so that the observer can again track the behavior of plant properly, and the system can continue operation in a degraded, but satisfactory level.

III. MODELING PARADIGMS

The FACT approach is model-based, so the designer's focus is on building models that correspond to their understanding of the system, rather than on executable program code. The Reconfigurable Controller and the Fault Adaptive Control Units are automatically generated from these models of the plant, controllers, the reconfiguration procedures, and the plant interface. Different tasks require different models, but some tasks may use models at different levels of abstraction achieving results at different granularity levels and with different computational complexity. In this section, the modeling paradigms currently used in the system are presented.

A. Hybrid Bond Graphs–Plant Models

We use bond graphs as the modeling paradigm in the continuous domain [3]. Bond graphs represent energy-based models of the system in terms of the effort and flow variables of the system. Bonds represent interconnections between elements that exchange energy, and they are described by two generic variables, *effort* and *flow* whose product defines the rate of flow of energy (power). Bond graphs represent a generic modeling language that can be applied to a multitude of physical system domains, such as electrical, fluid, mechanical, and thermal systems. An example is shown in Fig. 2. State-space equations used in the Hybrid Observer and Parameter Estimation Units can be systematically derived from the bond graph model of the system. In addition, temporal causal graphs (TCGs) can also be systematically derived from bond graphs, and they are used in the Hybrid Diagnosis Unit. An enhanced form of bond graphs, called Hybrid bond graphs (HBGs) [4] introduce controlled junctions that facilitate the modeling of discrete mode transitions (i.e., reconfigurations) in the system topology.

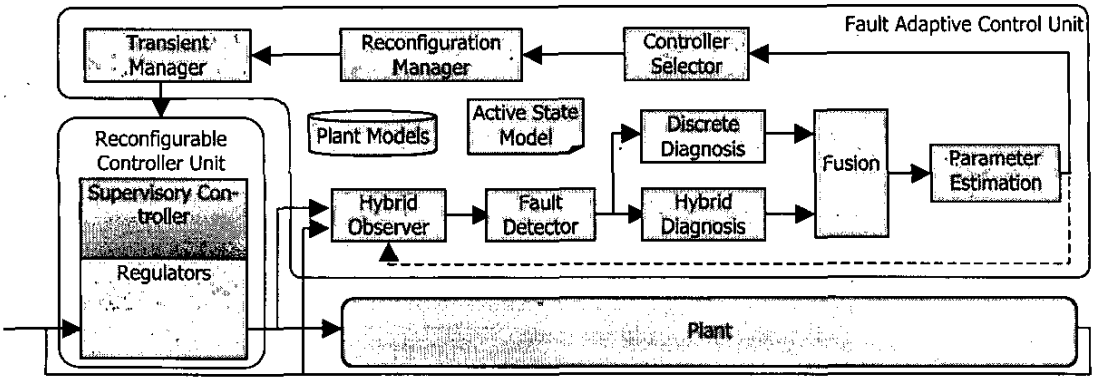

Fig. 1. Fault Adaptive Control Architecture

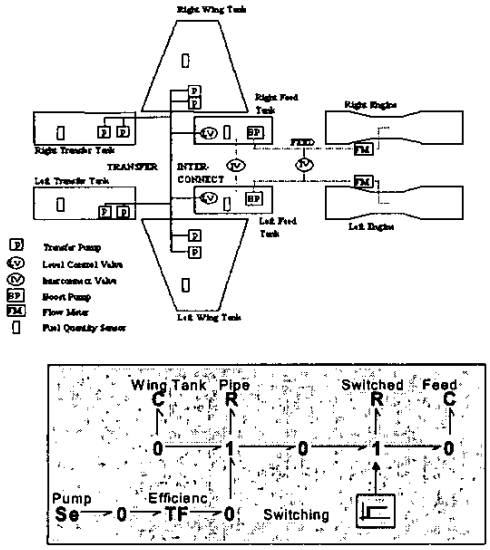

Fig. 2: Fuel Transfer system and partial HBG model

B. Hybrid Automata–Hybrid Observer

Our system model for the plant is a hybrid automaton that combines finite state machines with continuous representations [5], [1]. We have derived a transformation scheme for building hybrid automata models from the HBG models created by the modeler. The FSM, whose states correspond to the modes of operation of the system, captures the possible mode transitions in the system. A continuous system model that governs behavior evolution in that state augments each FSM state.

In a system containing N binary switching elements (i.e., N controlled junctions in the HBG) the total number of modes is 2^N, which may be large enough to make it infeasible to exhaustively generate the complete hybrid automaton. We avoid this computational problem by enumerating states of the hybrid automaton (modes of system operation) dynamically as system behavior evolves. The actual dynamical model of the system is generated in the form of state-space equations from the Bond-graph model in a particular mode of the system (Fig. 3). The Active State Space Equations are regenerated at every mode-change, but the computational complexity is reduced by caching previously generated mode-equation pairs. The State Space Equations represent linear systems, however, in the system the parameters can be recomputed at every time-step, so a piecewise linear approximation of a nonlinear system is straightforward. Typical nonlinear elements in our system are pipes and valves whose resistance parameters are dependent on the actual flow.

C. Models used by the Fault Detector

The Fault Detector employs simple models of the system and signals for robust detection of discrepancies between expected and measured behavior of the plant. The discrepancies are defined by *residuals*, i.e. differences between the measured and predicted signals. The deviation of the residuals from its ideal zero value is a measure of discrepancy. To handle modeling errors and measurement noise, the Fault Detector uses a statistical test (approximate z-test [6]) to determine the significance of the deviation. To perform this task we employ a standard model, where (i) the measurement noise is white, Gaussian with zero mean and constant (estimated) variance, (ii) an upper bound of the modeling error is known, and (iii) sensor accuracy for each signal is known. Based on these assumptions the modified z-test can be performed in the following way:

The variance of each signal is estimated in a large moving window (preceding the fault occurrence). This is used in the z-test to determine whether the mean value of a signal calculated in a small moving window significantly differs from zero. The confidence level and the window sizes used in the estimations are parameters provided by the system designer.

D. Temporal Causal Graphs–Hybrid Diagnosis

Our method for hybrid diagnosis is based on a qualitative approach for analyzing the transients that correspond to an abrupt change in the parameter value of a component. We briefly outline our method for analyzing transients in the continuous time domain, and then discuss its extension to the hybrid domain, where discrete mode changes cause the model of the system to switch.

Fault isolation from transients is based on the Temporal Causal Graph (TCG) that is derived from the Bond graph model of the system. The TCG captures causal relations between the system variables, in the form of a directed graph. The vertices of the graph represent the effort and flow variables in the system, whereas the links represent the cause-effect relations among the variables. The labels assigned to the TCG links capture algebraic and temporal relations. Fur-

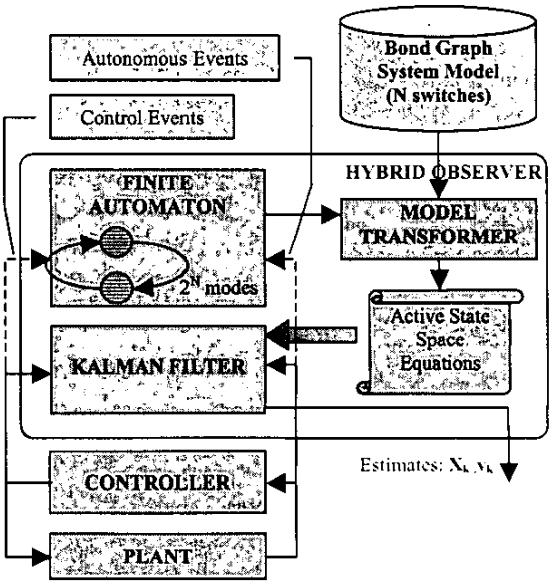

Fig. 3. The Hybrid Observer

thermore, the labels also explicitly include information about the system parameters, i.e., the dissipative (resistance), proportional (transformer and gyrator coefficients), and energy storage (capacitor and gyrator) elements that govern the relations between system variables. Mosterman and Biswas [7] describe in detail the algorithms used to generate fault candidates, their temporal signatures, and progressive monitoring for tracking the transient after fault occurrence to establish the true fault candidate. Sometimes the fault isolation scheme may not succeed in reducing the set of hypothesized fault candidates to a unique one, but we have shown that the combination of a parameter estimation scheme with our qualitative approach can be successfully employed to generate unique fault candidates [8].

The fault isolation scheme, extended to hybrid diagnosis, involves two additional steps: (i) *qualitative roll-back*, and (ii) *qualitative roll-forward*. The roll-back algorithm considers possible delays in fault detection, and because the fault may have occurred in previous modes, it goes back in the mode trajectory and creates hypotheses in previous modes using the observer estimated mode trajectory. During the crossover from a mode to a previous mode, the symbols are propagated back across the mode change. The hybrid hypotheses generation algorithm returns a hypotheses set, which includes the mode in which the fault is hypothesized to have occurred. Once a fault is hypothesized in a previous mode, a quick roll-forward method is implemented to enable progressive monitoring in the current mode. However, fault occurrences may change the parameters of the functions that determine autonomous transitions, therefore, the observer mode predictions are no longer correct. Mode transitions are also hypothesized, and this causes branching behaviours in the progressive monitoring scheme. Details of the hybrid diagnosis algorithm are presented in [9].

E. Timed Failure Propagation Graphs–Discrete Diagnosis

Timed failure propagation graphs (TFPG) [10] are causal models that describe the system behavior in the 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.

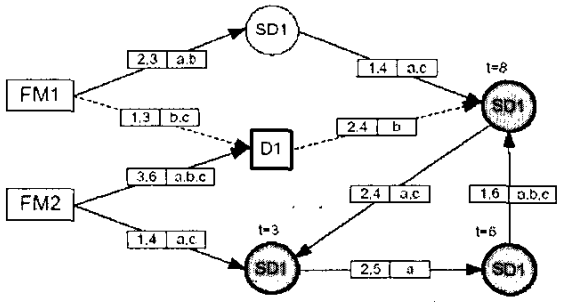

Figure 4: A hybrid failure propagation graph

An extended version of TFPG model, referred to as hybrid failure propagation graph [11], was 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 end, edges in the graph model can be constrained to a subset of the set of possible operation modes of the system. An example of a hybrid failure propagation graph is shown in the above figure.

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.

F. Controller Models

In the FACT architecture, the control component represents all the traditional control functions in an application. Although this component is implemented mainly in software, some components might utilize dedicated hardware components. This component is also "reconfigurable": its subcomponents, their parameters, and their interconnection can be changed during system operation. To represent this reconfigurable control component we have developed a modeling language, called Controller Modeling Language (CML) using the Model-Integrated Computing approach [12]. CML represents controllers on two levels: *regulatory* and *supervisory* levels. The designer supplies two models describing the structure and behavior of the controller:

Structural Model: The input and output ports of the controller entity are defined.

Behavioral Model: The controller behavior is described in two levels. In the supervisory level a segment of a parallel state machine is defined in each controller entity, the states of which represent the modes of operations (thus the 'Supervisory Controller' is distributed among the controller entities). The state transitions are governed by events and guards are generated from internal events and external (input) signals. Each state is associated with actions that are executed on entry, exit, or continuously while the state is active. Transitions can also have actions. Each action is associated with one or more scripts that describe the regulatory-level behavior of the controller.

G. Reconfiguration Models

The Reconfiguration Model gives the description of different controller configurations and their associated transitions. Each Reconfiguration Model contains a state machine and a set of input ports, output ports, controller blocks, and connections. The structural reconfiguration on the modeling level is specified by associating states with the I/O ports, controllers and connections that have to be active in the particular state. The state machine may also modify parameters and state variables, thus providing a means for non-structural reconfigurations as well.

H. Plan-Controller Interface Models

The Plant-Controller Interface model is a structural model that defines the connections between the controllers and the plant. It also contains some runtime environment specific details (e.g. clock rates, etc).

IV. EXAMPLE

We describe the application of the FACT technology to a real world example. We designed a fault-adaptive controller for the fuel system of an aircraft, whose schematic appears in Figure 2. The fuel system must provide an uninterrupted supply of fuel for the engines while maintaining the center of gravity of the aircraft.

The control system was developed using model-integrated computing tools [12]. The toolset developed was based on GME and a set of run-time components. Plant and controller models were built in this modeling environment, from which the implementation code was synthesized. When integrated with the generic FACT run-time components, the architecture for a specific application domain was instantiated.

Figure 4 shows the highest level of the hierarchical Bond Graph Model of the aircraft fuel system. On this level the blocks represent tanks, pipes, and valves. These elements are defined inside the blocks with higher detail. A controller model example is shown in Figure 5. The left hand side depicts the Structural Model with the input and output ports. The associated Reconfiguration Model block is also shown. The right hand side shows a simple Behavioral Model with

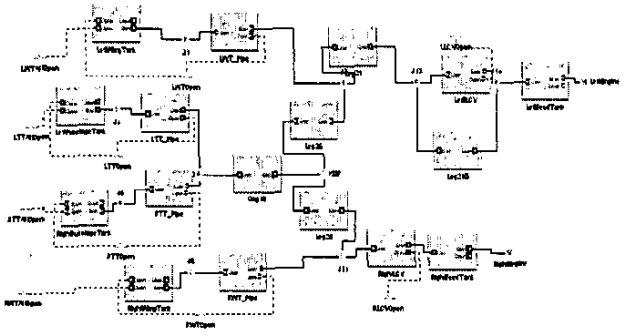

Fig. 4. The highest level of the fuel system Bond graph model

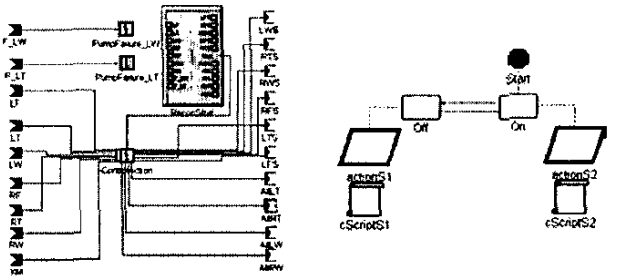

Fig. 5. A Controller Structure Model with the associated Reconfiguration Model (left) and a Controller Behavior Model (right).

two states, each associated with one action, and each action in turn associated with one script (the last association is visible on a different view of the MIC tool). The Plant-Controller Model and a portion of the TFPG Model are shown in Fig. 6. Figure 7 illustrates the inner structure of a Reconfiguration Model. Note that only the active components corresponding to the selected state are shown, the others are dimmed.

The operation of the system is illustrated in Figure 8 and Table 1. Two faults were injected: at time 100s the Left Wing Tank Pump degraded by 33%, while at time 800s the Left Fuselage Tank Pump degraded by 66%. Table 1 summarizes the TCG and TFPG diagnosis actions for the first fault. At the point of fault occurrence, a discontinuity was detected in the transfer manifold pressure, which resulted in 10 potential candidates for the TCG, and 5 for the TFPG. As more signals deviated, more symbols were generated for the TCG and the

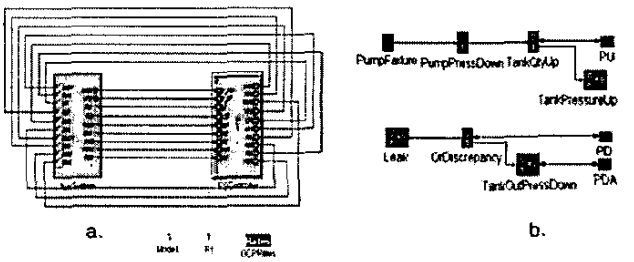

Fig. 6. a. The Plant-Controller Interface Model. b. A portion of the time Failure Propagation Graph for tank pump failures.

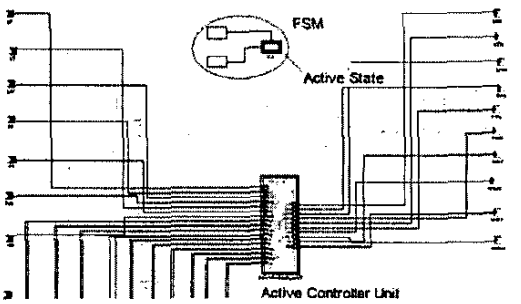

Fig. 7. Reconfiguration model. This view shows the active ports and controllers associated with a state of the controlling Finite State Machine. Inactive elements are dimmed.

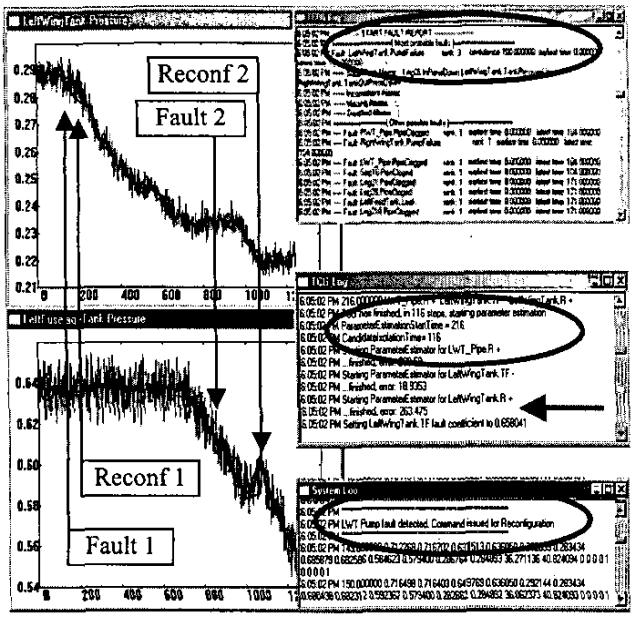

Fig. 8. Fault detection and reconfiguration. The noisy measurements and the observer's predictions are shown on the plots. Two faults occurred (at time instants 100 and 800). Both were detected and the appropriate reconfigurations were made. The logs (TFPG, TCG and System) correspond to the first fault.

more alarms were activated and reported to the TFPG reasoner. (As indicated, the diagnosis scheme worked through a number of mode changes.) By time point 159, the TFPG reasoner reported a high rank (=3) for the true failure mode Left Wing Tank Pump failure, and the TCG diagnosis reduced its candidate set to three at time point 167. At time point 216 the qualitative diagnosis was terminated without further improvement. The parameter estimation was initiated after the termination of the qualitative diagnosis, with 3 candidates. It successfully determined the true fault (Left Wing Tank Pump failure), and the magnitude of the parameter change (0.658). After the reconfiguration, the system continued to operate, as seen in the tracking of Fig. 8, but a second fault occurred at time point 800. This fault was also correctly detected and the appropriate reconfiguration was made. The TCG and the TFPG logs illustrate the identification of the first fault, and the System log shows the reconfiguration actions.

V. CONCLUSIONS

Model-based fault adaptive control architecture was proposed for complex dynamic systems. The FACT architecture is able to detect, isolate and identify faulty elements and modify the control strategy so that the operation is maintained. The proposed solution utilizes several aspects of system models, from which the FACT architecture can be synthesized, using Model-Integrated Computing Techniques. The modeling paradigms and the basic building blocks were presented, and the operation of the system was illustrated through a real world application example.

Table 1: Trace of Fault Isolation and Estimation
Left Wing Tank Pump Degradation

Time	Measured Deviation	Fault Set
100	Transfer Manifold Pressure (XMP)	13 fault candidates
101	XMP: discontinuous	10 candidates (TCG) 5 candidates (TFPG)
106 144	*Mode Change: Left Feed Tank On* *Mode Change: Left Feed Tank Off*	
147	LeftWing Tank Pressure ++	4 candidates (TCG))
149	*Mode Change: Left Wing Tank – Second Pump On*	
159	Left Feed Tank Pressure --	1 candidate (TFPG)
167	Left Feed Tank Pressure --	3 candidates (TCG)
216	*Mode Change: Left Feed Tank On*	
216	Parameter estimation	**LeftWingTank.TF fault coefficient: 0.658**

ACKNOWLEDGMENTS

The research was sponsored by DARPA/IXO SEC program (F30602-96-2-0227) and NASA IS NCC 2-1238. We would like to thank Dr. Kirby Keller and Mr. Tim Bowman of the Boeing Company for their support.

REFERENCES

[1] Alur, R. et al., "*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, 1993.

[2] Simon, G., T. Kovácsházy, G. Péceli, "*Transient Management in Reconfigurable Systems*," in P. Robertson, H.Shrobe, R.Laddaga (Eds): "Self Adaptive Software (First International Workshop, IWSAS 2000, Oxford, UK, April 17-19, 2000, Revised Papers), Lecture Notes on Computer Science 1936". Springer, New York, pp.90-98, 2001.

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

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

[5] Branicky, M.S., V. Borkar, S. Mitter, "*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, 1994.

[6] Kirk, R.E. Statistics: An Introduction, Wadsworth Publishing, 1999.

[7] Mosterman, P.J. and G. Biswas, "*Diagnosis of Continuous Valued Systems in Transient Operating Region,*" IEEE Transactions on Systems, Man, and Cybernetics,.vol. 29, pp. 554-565, 1999.

[8] Manders E.J., S. Narasimhan, G. Biswas, and P.J. Mosterman, "*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, 2000.

[9] Narasimhan, S. and G. Biswas. *An Approach to Model-Based Diagnosis of Hybrid Systems:* in *Hybrid Systems: Computation and Control (HSCC '02)*. 2002. Stanford, CA: Springer Verlag.

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

[11] S. Abdelwahed, G. Karsai, and G. Biswas, "*Robust Diagnosis of Switching Systems*," 5th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, Washington D.C., 2003.

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