A Model Integrated Computing Tool-Suite for Fault-Adaptive Control

Eric-J. Manders, Gautam Biswas, John Ramirez, Nagabhusan Mahadevan, Wu Jian
and **Sherif Abdelwahed** [1]

Abstract. The proliferation of safety-critical embedded systems has created great demands for online fault diagnosis and fault-adaptive control techniques. A number of methodologies have been proposed, but the implementation of on-line schemes that integrate the fault detection, isolation, identification, and fault accommodation or reconfiguration tasks remains challenging. We present a tool chain using a Model Integrated Computing (MIC) approach to develop Fault adaptive control systems. The domain specific features support physical system behavior modeling using the Hybrid Bond Graph paradigm. This, combined with models of the controller, fault detection, isolation, identification, fault-adaptation and reconfiguration schemes provides the basis for developing the run-time online computational system. The key component of the runtime system is an *active state model* representation that is dynamically updated as mode changes occur in the system. FDI is realized through the Hybrid TRANSCEND scheme, and a decision theoretic approach to fault adaptivity is being developed. An additional feature of our system includes a simulation system automatically generated from the system model that allows for experimentation with a range of fault scenarios. We illustrate the work on a water recovery system application.

1 Introduction

The increasing complexity of embedded systems and their use in safety-critical applications has imposed strict requirements on their reliability, robustness, and availability. These concerns are guiding the development of model-based approaches to diagnosis, and they include a wide range of solution paradigms that range from traditional signal analysis and control systems approaches [8] to the use of artificial intelligence techniques [9]. A particular solution may emphasize an aspect of the diagnosis problem, i.e., fault detection, fault isolation, and fault identification (FDI^2), and a comprehensive diagnosis architecture may combine multiple design approaches [8].

However, to achieve autonomy, the overall objective must extend beyond FDI to the task of invoking actions online during operation that mitigate the effects of the fault. This makes the system design problem quite complex because they may combine controller adaptation or system reconfiguration to compensate for or to eliminate the effects of the fault. The online scheme for FDI^2 and fault recovery/accommodation requires a number of component modules that are by themselves complex, and their composition results in a very complex run time computational system that has to meet a number of performance guarantees. Runtime diagnosis architectures have been proposed in the literature (e.g., [8]), but there are few systematic approaches to developing tool chains that support generating of runtime systems for the given architecture. Further, there are few tools that provide support for exploring and analyzing the design space with the goal of coming up with good modular and integrated system designs. These notions are particularly important when the FDI^2 and fault adaptive control system are realized as an embedded system application, where the computational system has to be tightly coupled with the actual physical plant.

This paper describes an approach to building such a tool chain using a Model Integrated Computing (MIC) [12] approach developed at the Institute for Software Integrated Systems (ISIS) at Vanderbilt University. The tool chain, called *Fault Adaptive Control Technology* (FACT), implements our approach to modeling, diagnosis and fault-adaptive control. The tool set comprises of three primary aspects: (*i*) an environment for building dynamic models of the physical plant, its interface hardware that includes sensors and actuators, and the controller; (*ii*) Computational environments and run-time support for FDI^2 and fault adaptive control as an embedded system application; and (*iii*) a simulation environment based on the plant, interface, and controller models that allows for running simulation experiments of nominal and faulty system scenarios. This provides a powerful tool for testing system performance under different fault scenarios.

The implementation of FACT is a work in progress and currently covers a set of specific techniques for building the different run-time components of the fault-adaptive control system. System behavior is modeled using a component-oriented, compositional modeling scheme using the underlying Hybrid Bond Graph (HBG) modeling paradigm [14] to build the physical process models. A hybrid observer scheme based on the HBG model, combines extended Kalman filter schemes with hybrid automata to track system behavior. Online FDI is built upon an extension of the TRANSCEND approach for qualitative hybrid diagnosis coupled with quantitative parameter estimation for fault identification [3, 13]. The FACT tools have been evaluated on several real-world applications [15]. In this paper, we focus on the specifics of the design and implementation of the tool chain, and describe its application to a Water Recovery System, a component of an Advanced Life Support system for extended duration human space exploration [7].

Section 2 gives a brief introduction of the water recovery system. Section 3 introduces the modeling tools, section 4 describes the simulation support, section 5 the run-time system support, and section 6 presents conclusions of this work.

[1] Department of Electrical Engineering and Computer Science, and the Institute for Software Integrated Systems, Vanderbilt University, P.O. Box 1824 Station B, Nashville, TN 37235-1592 USA. Phone: +1 615 322-0732, fax: +1 615 343-6702. (e-mail: {gautam.biswas}@vanderbilt.edu)

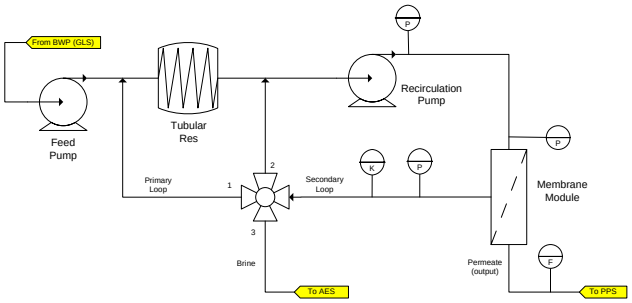

Figure 1. Process engineering diagram of the Reverse Osmosis subsystem.

2 The Advanced Water Recovery System

An actual Water Recovery System (WRS) testbed was designed and built at the NASA Johnson Space Center (JSC), and a detailed description of this system may be found in [4, 16]. Throughout this paper, we focus on the Reverse Osmosis (RO) subsystem of the WRS. Figure 1 shows its process engineering diagram [7, 16].

The RO system receives waste water from the Biological Waste Processor (BWP) after organic compounds have been removed from it. This water is pushed at high pressure through a membrane, the key functional component [16] of the system. Clean water, passes through to the Post Processing system (PPS), and the remaining water is recirculated in a feedback loop. As a result, the concentration of impurities in the recirculating water increases with time. The water that is recirculated will eventually be transferred to an Air Evaporation System (AES) for additional treatment.

The system cycles through three operating modes, which are determined by the position of a multi-position valve. The feed pump, which is always on, pulls effluent from the BWP and creates a flow into a tubular reservoir (coil). In the *primary* mode the input flow into the system is mixed with the recirculating water (recirculation loop). The recirculation pump boosts the liquid pressure and pushes it into the membrane. A transition to the *secondary* mode occurs after a predetermined time interval. In secondary mode the recirculating fluid is directly fed back to the membrane in a smaller loop to increase flowrate and maintain sufficient flow through the membrane. Membrane resistance increases as it accumulates dirt over time. The outflow of clean water from the loop causes an increase in brine concentration in the water remaining in the loop, and at a predetermined point that corresponds to 85% of volumetric recovery of water, a transition is made to the *purge* mode where the recirculation pump is turned off, and concentrated brine is pushed out to the AES subsystem. The system then cycles back to the primary mode. A complete cycle (modes 1 through 3) takes approximately four hours.

The actual physical system has been extensively instrumented. Figure 1 shows the five measured variables that are used for diagnosis in the current work: (*i*) the pressure immediately after the recirculation pump, P_{pump}, (*ii*) the pressure of the permeate at the membrane, P_{memb}, (*iii*) the pressure of the liquid in the return path of the recirculation loop, P_{back}, (*iv*) the flow of the effluent, F_{perm}, and (*v*) the conductivity of liquid in the return path of the recirculation loop, K. The sampling time on the sensors is assumed to be 120 (s).

3 The FACT modeling paradigm

The design of the tool chain follows a Model Integrated Computing (MIC) approach. Key elements of a MIC-based approach are:

1. a *Domain Specific Modeling Language* (DSML) defined using a meta-programming environment,
2. a *Domain Specific Modeling Environment* (DSME), that is created by instantiating the DSML in a meta-programmable modeling environment, and
3. *model translators/interpreters*, that generate analysis tools and synthesize software components.

The DSML for the FACT approach, referred to as the FACT modeling paradigm [11], is being developed using the meta-programmable Generic Modeling Environment (GME) application [12].

The FACT paradigm supports three types of models: (*i*) Plant models, that define the dynamic behavior and describe the interface of a physical system, (*ii*) Controller models, that describe the algorithms used to operate the plant, and (*iii*) System models, that compose the plant and controller models.

The user constructs a model of the application through a compositional, component-oriented approach. The use of component libraries can facilitates model reuse when available. Components are connected through input and output ports, and a constraint manager prevents the user from making modeling errors by dis-allowing unintended connections. Furthermore, models can be constructed through hierarchical decomposition allowing multiple abstraction levels.

To reduce visual clutter for the modeler, a model type may have multiple *viewing aspects*. For the plant model, the primary aspects are the system dynamics (HBG), and Input/Output aspect. The controller model defines a finite state machine aspect and a reconfiguration aspect. Plant and controller models are discussed in detail in the next section.

There are currently two model interpreters in the paradigm: (*i*) an interpreter that generates a MATLAB/Simulink simulation model of the system, and (*ii*) an interpreter that generates the model representation used to instantiate the run-time FDI^2/fault adaptive control system. It is this latter representation that is used to create model representations for tracking system behavior, quantitative and qualitative diagnosis components, and parameters to configure run-time algorithms.

3.1 The Plant Model

3.1.1 Systems Dynamics Aspect: Hybrid Bond Graphs

The component oriented, hierarchical modeling approach in the paradigm supports the creation of intuitive behavioral models for the system, that can be tailored to the diagnosis and control application. Multiple levels of abstraction allow the modeler to follow the engineering design of the physical system that is being modeled. At the lowest level in the behavior model, the components are represented as Hybrid Bond Graph (HBG) fragments. Hybrid Bond graphs (HBG) are an extension of the bond graph formalism that allows some elements to have discrete states, giving the modeler the ability to create domain-independent models that can describe both continuous and discrete behaviors of a system [14].

Components in the plant model can be connected through three types of ports: (*i*) decision ports communicate discrete information about modes of operation, (*ii*) energy ports handle the transfer of energy, and (*iii*) double valued ports, pass numeric information (signals) to sensors or other functions in the HBG.

Mode changes in the system add discrete components to continuous system behavior. They may be attributed to control actions or autonomous changes, i.e., when state variables of the system exceed pre-specified threshold values. In hybrid bond graphs, mode

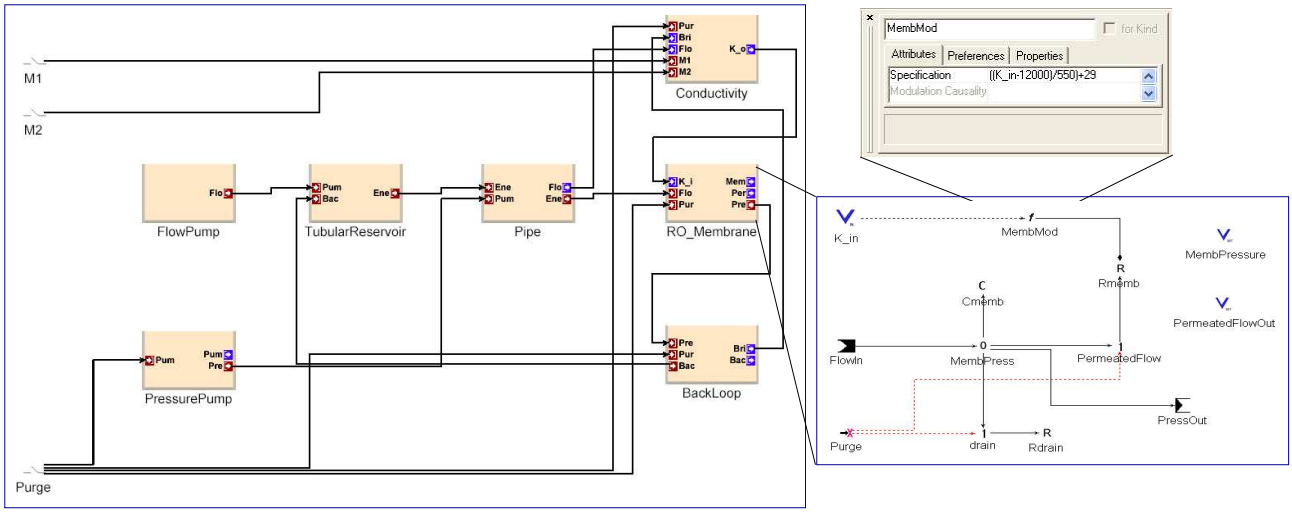

Figure 2. FACT paradigm model of the RO system at the highest level, showing the HBG model view. The refinement model of the `RO_membrane` component is a flat HBG model. The attribute editor for the modulation function `MembMod` shows the algebraic relation on the `K_in` input signal.

changes are reflected by junctions that turn *off* and *on* based on the value of switching signals. This is achieved through *decision functions*, that compute the switching signals as a function of system variables. Nonlinear systems are modeled by components that have time-varying parameters, i.e., their parameter values are functions of system variables. The functions capturing the nonlinear behaviors are called *modulation functions*.

Figure 2 shows the top level model of the RO system in the HBG aspect. The components of this system appear as blocks at this level. All components are represented at the lowest level of the hierarchy by HBG model fragments. As an example, we show the HBG fragment of the `RO_Membrane`. All three types of ports are used in this HBG component, a double value port (`K_in`), a decision port (`Purge`) and an energy port (`FlowIn`). The HBG for the `RO_Membrane` component also has a modulating function that defines the value of RO_Resistance `Rmemb` as a function of the water conductivity `K_in`. The attribute pane for the modulating function element is shown. The specification field shows the actual function.

3.1.2 I/O Aspect: Sensors and Actuators

The Input/Output aspect of the plant defines how the connection between plant components and actuators and sensors. Sensors measure plant variables associated with junctions in the HBG model. A sensor connected to a one-junction will read the flow value at that junction, while a sensor connected to a zero-junction will read the effort value at that junction. Actuators may be either continuous or discrete. Sensors and actuators may also have attributes associated with them.

3.2 The Controller model

The controller's task is addressed at two levels. At the supervisory (discrete) level, reconfiguration implies modification of high-level control actions, such as changing pump speeds, and turning valves and pumps on and off. 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. The Control aspect aspect supports reconfigurable control designs. Control switching is specified using a finite state machine representation, and actual controllers are linked in as resources in attribute descriptions of control components.

In the FACT architecture, the control component is implemented in software. The designer supplies two models: (*i*) the *structural model*, which is based on a run time computational architecture, and specifies the interconnections between the plant and the controllers through the I/O interfaces, i.e., sensors and actuators; (*ii*) the *behavioral model*, that captures low level controller behavior as a state machine. The state transitions in the state machine are governed by events (timers) and/or guard conditions generated from internal events and external (input) signals. External events may include events generated by the FDI^2 system. Each state may be associated with one or more actions that are executed on entry, exit, or while continuing to remain in that state. At the supervisory control level, *reconfiguration strategy models* describe the different configuration (modes of operation) of the controller, and the possible transitions between the configurations. This is described as a parallel state machine. In section 5, we demonstrate an online fault-adaptive reconfigurable decision-theoretic controller applied to the RO system.

4 Experiments with Fault Scenarios

Simulation tools are essential for developing the right models of complex systems. Through an iterative process, system behavior generated by simulating the models allows the the designer to refine the models by comparing against actual system measurements, and then using parameter estimation techniques to improve model accuracy. The simulation environment provides added functionality in that it allows modelers to insert parametric faults into system components at user-specified times during system operation, with a chosen fault profile and fault magnitude. This provides a powerful tool for testing fault-adaptive performance of the system in a simulation environment.

The model interpreter constructs an abstract representation of the HBG model, and then synthesizes a new model representation for a particular simulation environment, currently MATLAB/Simulink. However, other implementations may target alternate simulation

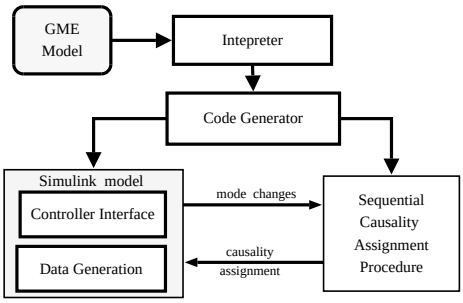

Figure 3. Mapping the GME model to a Simulink model.

models, such as Modelica [5]. A Graphical User Interface allows easy scenario construction, and manual editing of the MATLAB/Simulink model is not required.

The simulation consists of two main components: (*i*) the Simulink block diagram, and (*ii*) a causality assignment (SCAP) algorithm. These two components contain all the information described in the Hybrid Bond graph as well as the Input/Output aspect information of the model. Figure 3 illustrates the mapping of the GME model to the Simulink model.

HBG components are implemented as a library of Simulink blocks. The Simulink model preserves the component-based hierarchy of the system model. To facilitate inspection of the generated model, all layout information from the GME model is retained. Figure 4 illustrates the top-level MATLAB/Simulink model for the RO system, showing the controller, the system and the data plotters for visualization.

Figure 5 shows the mathematical representation of a capacitor component created by using blocks from the predefined library of the conversion package. Similarly, junctions, which are multi-port bond graph elements translate to summation blocks in the Simulink model. The mathematical relations captured by the junctions are algebraic, and they depend on the junction type (common effort and common flow) and the direction of the connecting bonds. The connecting bonds establish the energy flow paths among the connected elements. The causal structure established by the SCAP algorithm, described below, establishes the order in which variable values are calculated using the algebraic relations.

The notion of switching junctions is preserved in the simulation model. Junction blocks may be enabled/disabled which effectively connects and disconnects all connected components. Mode switching

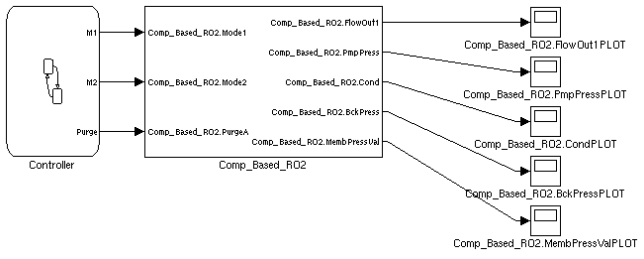

Figure 4. Top-level MATLAB/Simulink model of the RO system as generated by the interpreter, showing the controller block, the system model block, and components to display simulation data.

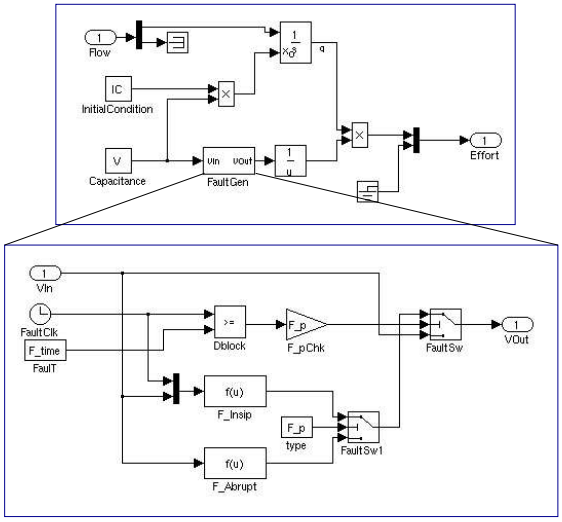

Figure 5. Simulink model of a capacitor, with `FaultGen` block detailed. Fault profile and time of fault occurrence are parameters to this component.

requires the simulator to recompute the causal relations among the variables in the new mode, and then update the junction blocks with this information. The causality assignments are computed based on the Sequential Causality Assignment Procedure (SCAP) [10] after every mode change. Junctions use the updated causality assignments to determine the order in which the system variables must be solved to allow efficient simulation.

5 On-line Model-based FDI^2 and Fault adaptivity

The run-time FACT system is configured through the model representations generated by the GME-translation step. At system startup this is used to build a dynamic representation of the controlled physical plant, as a hybrid system with reconfigurable control. It is also used to configure the FDI^2 functionality and fault adaptive behavior.

The architecture of the FDI^2 part of the system, the Hybrid extension of TRANSCEND, is shown in Figure 6. The model-based approach combines robust tracking of nominal system behavior using extended Kalman filter techniques [6], statistical fault detection and symbol generation techniques, and fault isolation scheme that is based on the qualitative analysis of the system dynamics immediately after the time point of fault occurrence followed by quantitative parameter estimation to further resolve fault isolation if needed and also identify the fault [3]. The extension of these methods to hybrid systems incorporates both controlled and autonomous mode switches. The key components of the architecture are the active state model (ASM), the hybrid observer, and the combined FDI^2 components. Each of these components is supported by the run-time environment to create a highly configurable FDI^2 system.

5.1 Active State Model

The Active State Model (ASM) is a dynamic component that maintains the current model of the system at run time. It has two components: (*i*) the *structural model* that contains the HBG model of the system, and (*ii*) the *model parameters*, which is a data structure that contains the current value of all model parameters. Parameter values are updated either in response to mode changes, or upon completion of parameter identification in response to a fault. The ASM contains

all information to generate the alternative model representations in the system: (*i*) a state-space model used by the extended Kalman filters in the hybrid observer and the optimization algorithm for parameter estimation, and (*ii*) a Temporal Causal Graph (TCG) model used by the qualitative fault isolation algorithm.

5.2 Hybrid Observer

The hybrid observer includes an Extended Kalman filter, and a mode estimation component [11]. The estimated state is used to determine any autonomous mode changes. The estimated output variables are computed to allow the computation of a numerical residual as the difference between observed and estimated outputs.

5.3 Fault detection, Isolation and Identification

Through the sequence of fault detection, isolation, and identification an underlying data flow model defines the top-level computational model in the system.

Fault detection takes the numerical residual as input. The architecture allows for decoupling the fault detection from the symbol generation for the qualitative fault isolation scheme. The signal-to-symbol transformation component is realized as a bank of concurrent generators, one for each element in the residual vector. A single channel signal-to-symbol transformation, is realized as a filter bank, where each symbol is computed using FIR filters with quantization on the outputs [11].

Qualitative model-based fault isolation, the core of the TRANSCEND approach, implements a hypothesis generation and refinement scheme. During the hypothesis refinement phase, branching behavior maintains the valid hypotheses over all possible hypothesized modes for the faulty system. Finally, the quantitative parameter estimation phase is initiated after qualitative fault isolation cannot reduce the candidate set any further. This too, can be generalized to a generic fault identification scheme.

5.4 Decision Theoretic Control

An online adaptive control mechanism implements a resource management scheme using a decision-theoretic controller based on a multi-attribute utility function that models system performance: $V(p) = \sum_i V_i(p_i)$, where each V_i corresponds to a value function associated with performance parameter, p_i. The parameters, p_i, can

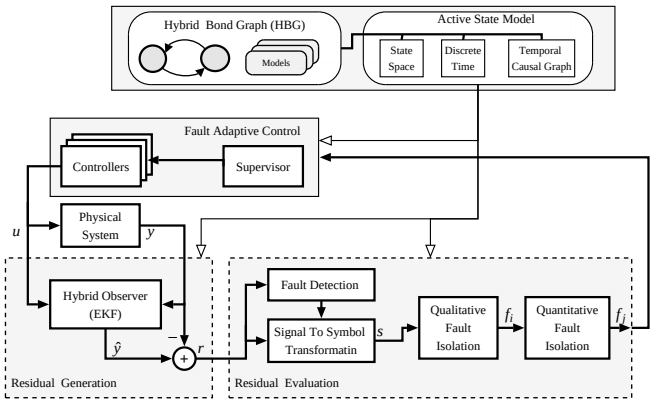

Figure 6. FACT run-time system architecture.

be continuous or discrete-valued, and they are derived from the system state variables, i.e., $p_i(t) = P_i(x(t))$. The value functions currently defined in our FACT paradigm are simple weighted functions of the form $V_i(p_i) = w_i * p_i$, where the weights take on values in the interval $[-1, 1]$, and represent the importance of the parameter in the overall operation of the system. The supervisory controller uses the active state model to predict possible behaviors corresponding to different action sequences for a finite forward time horizon, and then selects the action (i.e., control input) that maximizes the utility function. This process is then repeated for the next time step, and so on. Since the optimizing function operates on the current system model, the optimizing controller is fault adaptive. Decision making is adapted to the model with the newly estimated parameter value for the faulty component.

5.5 Fault scenario: Decrease in pump efficiency

For completeness we illustrate the operation of TRANSCEND on a fault in the RO system. A decrease in the efficiency of the recirculating pump is modeled by a decrease in the value of a bond graph component parameter of the pump, the gyrator, GY. We indicate this fault scenario as GY^-. Figure 7 shows the simulated plant data including the controller signals and the output of the observer, and the computed residual signals.

Table 1 shows the fault isolation results. Four qualitative fault isolation steps reduce the set of candidates from twelve model parameters down to three. Quantitative fault isolation and identification achieves the desired fault isolation result by eliminating the remaining spurious candidates and provides a good estimate (slight over estimation) of the actual fault size. On completion of the parameter estimation, the hybrid observer is updated with the new parameter values, and continues to track the new behavior, where the known changed system behavior becomes the nominal behavior. Complete diagnosis results for this system are presented in [2].

The online controller compensates for the fault by changing the mode switching pattern, and keeping the system in primary mode for a longer time in each cycle. The overall average utility after the occurrence of the fault decreased by only $.93\%$ for this example. The details of the fault adaptive behavior for this example are presented in [1].

6 Discussion and Conclusions

We have presented a MIC tool-suite for developing model-based FDI^2 and fault adaptive control systems that captures many of the general concepts that apply to this problem domain. The specific implementation of the FACT approach that represents our on-line diagnosis and fault-adaptive control schemes represents a reference realization of the archtecture.

Our long-term goal is to provide a computational framework that captures FDI^2 and fault accommodation concepts in a general enough manner, so that the framework may be used by researchers to implement their particular solutions in an efficient manner.

ACKNOWLEDGEMENTS

This work was supported in part through grants from the NASA-IS program (Contract number: NAS2-37143), DARPA SEC program (Contract number: F30602-96-2-0227), and the NASA-ALS program (Contract number: NCC 9-159). We acknowledge the help provided by Gabor Karsai and Gyula Simon and Vanderbilt in the development of the FACT architecture.

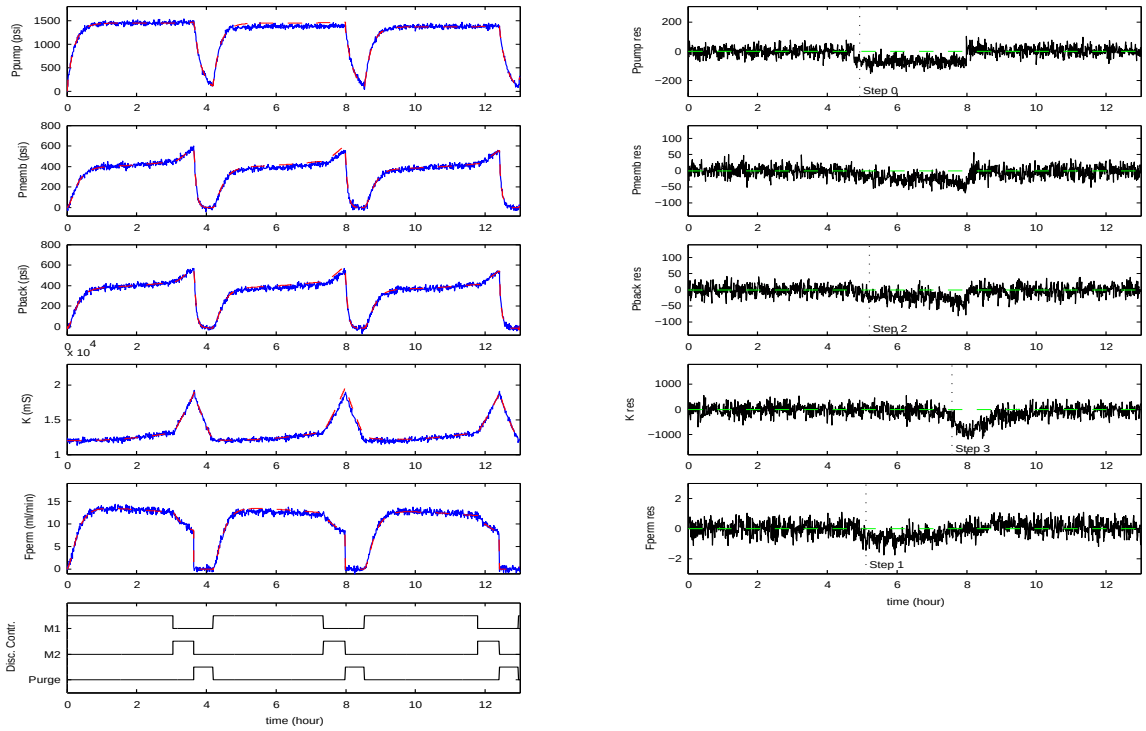

Figure 7. Simulation results for an abrupt fault GY^-, showing the data (*l*) and residual (*r*). Fault size 5%, occurring at $t_f = 18000\ (s) = 5(hours)$. (primary mode in the second cycle).

Fault	$t - t_f$	Step	Symbolic	Fault Hypotheses and final parameter estimation
GY^-, 5% t_f : 18000	200	0	$e37 : (-, \cdot)$	$C_c^+, C_{memb}^+, I_{fp}^+, I_{ep}^+, R_{brine}^-, TF^+, R_{pipe}^-, R_{memb}^-, C_k^+, R_{fp}^+, R_{ep}^+, GY^-$
	880	1	$f25 : (-, \cdot)$	$I_{fp}^+, I_{ep}^+, R_{brine}^-, TF^+, R_{fp}^+, R_{ep}^+, GY^-$
	1240	2	$e1 : (-, \cdot)$	$I_{ep}^+, R_{brine}^-, R_{ep}^+, GY^-$
	1960	3	$e35 : (-, \cdot)$	I_{ep}^+, R_{ep}^+, GY^-
				parameter estimation: GY^- changed by 0.934

Table 1. Diagnosis result for the GY^- fault.

REFERENCES

[1] S. Abdelwahed, J. Wu, G. Biswas, J. Ramirez, and E.-J. Manders, 'Online hierarchical fault adaptive control for advanced life support systems', in *Proc 32nd Int Conf Environmental Sys*, (2004). To Appear.

[2] G. Biswas, E.-J. Manders, J. Ramirez, N. Mahadevan, and S. Abdelwahed, 'Online model-based diagnosis to support autonomous operation of an advanced life support system', *Habitation: An International Journal for Human Support Research*, (2004). To Appear.

[3] G. Biswas, G. Simon, N. Mahadevan, S. Narasimhan, J. Ramirez, and G. Karsai, 'A robust method for hybrid diagnosis of complex systems', in *Proc. 5th IFAC Symp Fault Detection Supervision Safety Technical Processes*, pp. 1125–1131, Washington, DC, (June 2003).

[4] P. Bonasso, D. Kortenkamp, and C. Thronesbery, 'Intelligent control of a water recovery system: Three years in the trenches', *AI Magazine*, 19–43, (2003).

[5] W. Borutzky, 'Bond graphs and object-oriented modelling a comparison', *J Systems Control Eng*, **Vol 216 Part I**, 21–33, (2002).

[6] K. Brammer and G. Siffling, *Kalman-Bucy Filters*, Artec House, Norwood MA, 1989.

[7] B.E. Duffield and A.J. Hanford, 'Advanced life support requirements document', Technical Report JSC-38571, Rev. B, NASA-Lyndon B. Johnson Space Center, Houston, TX, (September 2002).

[8] P. M. Frank, S. X. Ding, and B. Köppen-Seliger, 'Current developments in the theory of FDI', in *Proc. 4th IFAC Symp Fault Detection Supervision Safety Technical Processes*, pp. 16–27, Budapest, Hungary, (2000).

[9] P. M. Frank and B. Köppen-Seliger, 'New developments using AI in fault diagnosis', *Engineering Applications of Artificial Intelligence*, **10**(1), 3–14, (1997).

[10] D. C. Karnopp, D. L. Margolis, and R. C. Rosenberg, *Systems Dynamics: A Unified Approach*, John Wiley and Sons, NY, second edn., 1990.

[11] G. Karsai, G. Biswas, T. Pasternak, S. Narasimhan, G. Peceli, G. Simon, and T. Kovacshazy, 'Towards fault-adaptive control of complex dynamical systems', in *Software-Enabled Control – Information Technology for Dynamical Systems*, eds., T. Samad and G. Balas, chapter 17, 347–368, Wiley-IEEE press, NJ, (2003).

[12] G. Karsai, J. Sztipanovits, A. Ledeczi, and T. Bapty, 'Model-integrated development of embedded software', *Proc IEEE*, **91**(1), 145–164, (January 2003).

[13] P. J. Mosterman and G. Biswas, 'Diagnosis of continuous valued systems in transient operating regions', *IEEE Trans. Syst., Man Cybern. A*, **29**(6), 554–565, (1999).

[14] P.J. Mosterman and G. Biswas, 'A theory of discontinuities in physical system models', *J Franklin Institute*, **335B**(3), 401–439, (1998).

[15] S. Narasimhan, G. Biswas, G. Karsai, T. Szemetzy, T. Bowman, M. Kay, and K. Keller, 'Hybrid modeling and diagnosis in the real world: A case study', in *Working Papers Thirteenth Int Workshop Principles Diagnosis*, (June 2002).

[16] K. D. Pickering, K. R. Wines, G. M. Pariani, L. A. Franks, J. Yeh, B. W. Finger, M. L. Campbell, C. E. Verostko, C. Carrier, J. C. Gandhi, and L. M. Vega, 'Early results of an integrated water recovery system test', in *Proc 29th Int Conf Environmental Sys*, (2001).