Hybrid Modeling and Diagnosis in the Real World: A Case Study

Sriram Narasimhan, Gautam Biswas, Gabor Karsai & Tivadar Szemethy
EECS Dept. & ISIS, Vanderbilt University
Box 1824, Station B, Nashville, TN 37235.
{nsriram, biswas, gabor, tiv}@vuse.vanderbilt.edu

Tim Bowman, Mark Kay & Kirby Keller
The Boeing Company
MC S111-1335, Box 516
St. Louis, MO 63166.
{timothy.e.bowman, mark.c.kay,kirby.j.keller}@boeing.com

Abstract

Applying model-based diagnosis techniques to systems that exhibit hybrid behavior presents an interesting set of challenges that mostly revolve around interactions of the continuous and discrete components of the system. In many real world systems, the overall physical plant is inherently continuous, but system control is performed by a supervisory controller that imposes discrete switching behaviors by reconfiguring the system components, or switching controllers. In this paper, we present a case study of an aircraft fuel system, and discuss methodologies for building system models for online tracking of system behavior and performing fault isolation and identification. Empirical studies are performed on detection and isolation for a set of pump and pipe failures.

1 Introduction

Most present-day systems that we use are designed to be repairable. Failures. either physical (hardware) or logical (software), and the resulting maintenance are a fundamental part of the economics of ownership. Fault diagnosis involves the detection of anomalous system behavior and the isolation and identification of the cause for the deviant behavior. When the system includes safety-critical components, failures or faults in the system must be diagnosed as quickly as possible, and their effects compensated for so that control and safety can be maintained. The term, *diagnostic capabilities*, refers to the abilities of a system to detect a failure and isolate it to a failed unit. Quick detection and isolation allows for quick corrective actions that may include reconfiguration of system functions to prevent damage and maintain control.

Fault accommodation requires tight integration of *online* fault detection, isolation, and identification with the system control loop that may be designed to take appropriate control actions to mitigate the effect of the faults and help maintain nominal system operation. Failure to detect faults reduces system availability, results in failed or incomplete missions, and, in some cases, may even lead to catastrophic failures that lead to loss and destruction of the system. Therefore, fault diagnosis is critical to achieving system performance and life-cycle cost objectives.

In general, systems are *dynamic*, i.e., their behavior changes over time. Faults impose additional *transients* on the dynamic behavior, but that may be difficult to detect and characterize, especially in the presence of model disturbances and noisy measurements. Moreover, in physical systems natural feedback from the system and controller actions may mask the transient behavior if they are not detected soon after they occur. This motivates the development and use of online model-based fault detection and isolation methods. Model-based techniques employ a model to predict nominal system behavior. The model must be constructed at a level of detail where system behavior can be mapped to system components and parameters. The relations in the model are employed to map observed deviations between measurements and values predicted by the model to possible faults in system components. Continued monitoring helps establish a unique fault or set of faults associated with the system.

Most real-life systems are equipped with a limited number of sensors to track system behavior, and analytic redundancy methods have to be applied to derive non-local interaction between potential faults and observations. These techniques have been applied to a variety of schemes used in the diagnosis of discrete [deKleer and Williams, 1987], discrete event [Lunze, 1999; Sampath *et al.*, 1996] and continuous systems [Gertler, 1997; Mosterman and Biswas, 1999]. The traditional approach to hybrid system diagnosis has been to use a single continuous model with complex non-linearities, or abstracting the continuous dynamics to a discrete event model. Complex non-linearities complicate the analysis and they may introduce numerical convergence problems. Discrete event abstractions lead to loss of critical information, such as fault transient characteristics. Further, methods to identify the set of events that describe both nominal and faulty behavior is often a computationally challenging task bringing to question the scalability of such approaches. Hybrid system analyses require the use of multiple models of the system. Recent approaches to hybrid system diagnosis have incorporated appropriate model selection and mode estimation techniques at run time to track faulty behavior and perform fault isolation [McIlraith *et al.*, 2000; Hofbaur and Williams, 2002; Narasimhan and Biswas, 2001; 2002].

Model-based diagnosis techniques can only be as good as the models upon which they are based. Incomplete and incor-

rect models cause problems with the tracking and fault isolation tasks. The tracking process may produce false alarms or worse missed alarms. In the first case, diagnosis is triggered when there is no fault in the system. In the second situation, diagnosis is not triggered and a fault may be missed. Fault isolation with incomplete and inaccurate models may also produce false candidates and miss true candidates. On the other hand, building models that include unnecessary detail may increase computational complexity making online processing infeasible. Therefore, a key task in model-based diagnosis is to build accurate models at the right level of detail. This paper focuses on the pragmatics of model building and fault isolation by performing a case study on the fuel transfer system of an aircraft.

2 Fuel System Description

High-performance aircraft require sophisticated control techniques to support all aspects of operation, such as flight control, mission management, and environmental control. An aircraft's fuel transfer system maintains the required flow of fuel to the engines through different modes of operation, while ensuring that imbalances are not created that affect center of gravity of the system. Fig. 1. illustrates a typical fuel system configuration. The fuel system geometry is symmetric and may be split into left side and right side arrangements. The overall system can be divided into two main sub-systems: (i) *the engine feed system*, and (ii) *the transfer system*. The feed system consists of a left and right engine feed tank. The tanks are connected through pipes with controlled valves so that fuel can be transferred between the tanks if a fault occurs in one of the tanks. A boost pump in each of the feed tanks controls the supply of fuel from the tank to its respective engine. The transfer system moves fuel from the two forward fuselage and the two wing tanks to the engine feed tanks. The intent is to keep the engine feed tanks near full at all times so that sufficient fuel is available on demand, and if failures occur in the transfer system there is still a significant amount of fuel available for emergency maneuvers. The fuel transfer sequence is set up in a way that maintains the aircraft's center of gravity. To achieve this, pumps located in the fuselage and wing tanks are are turned on in a pre-determined sequence to transfer their fuel to a common transfer manifold (set of tubes). The fuel exits the transfer manifold through level control valves into the feed tanks.

A wide variety of sensors may be included in the fuel transfer system. Fuel quantity gauging sensors determine the amount of fuel in a tank. Engine fuel flow meters determine engine fuel consumption. Pressure transducers measure the transfer and boost pump pressures. Position sensors determine the open and closed states of valves. Each sensor comes at a cost that is determined by its weight, reliability, complexity, and cost. Therefore, designers often try to minimize the number of sensors while ensuring that the required control can be achieved.

The transfer system control schedules the pump operation to match a pre-defined transfer sequence shown in Table 1. The unit of the amounts in the table is the *pound*. Initially one wing pump in each tank is turned on. When a feed tank

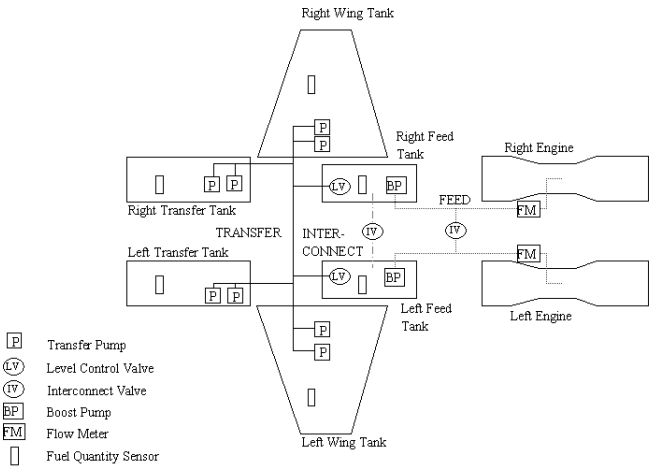

Figure 1: Fuel System Schematic

quantity decreases by *100 lbs*, the level control valve in that tank will be opened. The fuel then flows from the transfer manifold into the feed tank raising its level back to the full fuel quantity at which point the level control valve will be closed, stopping the fuel transfer.

Table 1: Fuel Transfer Sequence

Left Wing Tank	Right Wing Tank	Left Fuselage Tank	Right Fuselage Tank
2500	2500	3300	3000
2000	2000	3300	3000
2000	2000	3000	3000
1000	1000	2000	2000
0	0	1000	1000
0	0	0	0

The most common failures in this configuration are transfer and boost pump failures, and shutoff valve failures. The transfer pumps have two primary failure modes. One is a total loss of pressure caused by the impeller not turning. The other is a degraded state caused by mechanical wear, leakage, or electrical failure where the fuel flow rate falls below nominal values. The second failure can lead to the first condition over time. Faults in the boost pump mirror those in the transfer pumps. Valve failures are *stuck-at* conditions, i.e., their positions do not change even when they are commanded to do so. This can result from mechanical friction/jamming of the shaft or electrical failure of the motor or power source. In this work, we also consider partial failures of the valves. A third class of faults that we consider is leaks in the connecting pipes. Our goal is to develop an online diagnostic system for detection, isolation and identification of these faults.

3 Component-based Hierarchical Modeling for Diagnosis

Complex real-world systems are made up of a number of subsystems and components. Hierarchical component-based approaches, e.g., Statecharts [Harel, 1987], 20sim [van Amerongen, 2000], and Ptolemy [Buck *et al.*, 1994]. are a practical approach to constructing models of such systems, We have developed a new methodology for hierarchical component based modeling that customizes the graphical Generic Modeling Environment (*GME*) with a hybrid bond graph (*HBG*) approach for building hybrid models of physical systems with supervisory controllers. This section reviews our approach to hybrid bond graph modeling, then presents the *GME* methodology for building component-based models for the aircraft fuel transfer system.

3.1 Hybrid Bond Graphs

Our approach to modeling the fuel system is based on an extended form of bond graphs [Karnopp *et al.*, 1990], called *Hybrid Bond Graphs (HBG)* [Mosterman and Biswas, 1998]. Bond graphs present a methodology for energy-based modeling of physical systems. Generic bond graph components represent primitive processes, such as the energy storage elements, *inertias* and *capacitors*, and dissipative elements, *resistors*. Bonds represent the energy transfer pathways in the system. Junctions, which are of two types: 1 or *series*, and 0 or *parallel*, define the component interconnectivity structure, and impose energy conservation laws. Overall, the bond graph topology implies system behavior that combines individual component behaviors based on the principles of continuity and conservation of energy.

Extensions to hybrid systems require the introduction of discrete changes in the model configuration. In the *HBG* framework, discontinuities in behavior are dealt with at a meta level, where the energy model embodied in the bond graph scheme is suspended in time, and discontinuous model configuration changes are modeled to occur instantaneously. Therefore, the meta level describes a control structure that causes changes in bond graph topology using idealized switches that do not violate the principles of energy distribution in the system. Topology changes result in a new model configuration that defines future behavior evolution. To ensure physical principles are not violated, we have developed transformations that derive the initial system state in the new configuration from the old one. From this point on behavior evolution is continuous, till another discrete change is triggered at the meta level.

To keep the overall behavior generation consistent, the meta-model control mechanism and the energy-related bond graph models are kept distinct. The switching structure is implemented as localized *switched junctions* that provide a compact representation of the system model across all its nominal modes of operation. Instead of pre-enumerating the bond graph for each mode, the *HBG* uses individual junctions to model local mode transitions. The switched 0- and 1- junctions represent idealized discrete switching elements that can turn the corresponding energy connection on and off. A finite state machine determines the ON/OFF physical state of the junctions. The transitions in this automaton depend on both control signals and internal variable values.

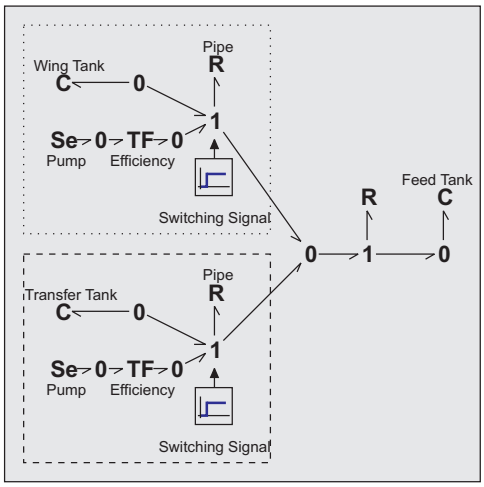

Figure 2: Hybrid Bond Graph Example

Fig. 2 shows the hybrid bond graph model of a portion of the fuel system. The dotted subsystem represents the wing tank, and the dashed subsystem represents the fuselage tank. In this simplified model, the tank system is modeled as a capacitor for storage of fuel, pump system as an effort source to boost the pressure and create an outflow, and pipes as conduits with resistive losses. For this configuration with two switched junctions, the system can be in four different modes. When the two junctions are off, there is no fuel supplied to the feed tank, one of the two tanks (wing or fuselage) can supply fuel to the feed tank, and both tanks may supply fuel to the feed tank at the same time. Switching of configurations is achieved by changing the switching signal values. Suppose the wing tank is supplying fuel, i.e., $signal_1 = 1$ (on) and $signal_2 = 0$ (off). To switch supplying tanks, we simply set $signal_1 = 0$ (off) and $signal_2 = 1$ (on). The state equation model for the new configuration can be easily derived online using a standard algorithm [Karnopp *et al.*, 1990].

3.2 GME

We have developed an approach for building component-based system models using a graphical modeling tool called Generic Modeling Environment(*GME*) [Ledeczi *et al.*, 2001]. *GME* is a configurable toolkit for creating domain-specific modeling and program synthesis environments. The configuration is accomplished through meta-models[1] specifying the modeling paradigm (modeling language) of the application domain. The modeling paradigm contains the syntactic, semantic, and visual presentation information of the domain, such as the concepts that form the building blocks for constructing models, the relationships among these concepts, how the concepts may be organized and viewed by the modeler, and the rules governing the composition of individual concepts and sets of concepts to form component and system

[1]The concept of meta-models in *GME* differs from the meta level switching models in *HBG*.

models. The modeling paradigm defines the family of models that can be created using the resultant modeling environment.

The meta-models specifying the modeling paradigm are employed to automatically generate the target domain-modeling environment, e.g., the *HBG* environment. The generated modeling environment is then used to build domain models that are stored in a model database. The primarily graphical, domain models can be conveniently stored in standard formats including *XML* to be used by specific applications. We have developed a *GME* modeling paradigm that describes the *HBG* modeling environment.

3.3 Hierarchical and Compositional Modeling in the fluid domain

Real life systems with embedded control tend to be complex, and system designers and engineers typically have a lot of difficulty in generating flat models of the entire system. Hierarchical and compositional modeling are useful tools that allow the system to be constructed in a structured way by modeling subsystems independently and composing them to generate system models. The two main steps in this approach are: (i) decomposition into subsystems and building models of subsystems, and (ii) specifying interactions between subsystems and using composition operators to build system models. This approach provides a number of advantages, such as simplicity in model building, independence in building subsystem models, and modularity and reusability of the generated components.

To model the fuel transfer system, we develop models of typical components in the fluid domain, such as tanks, pipes, and pumps. Pipes may include valves that regulate flow. Pumps and valves can be turned on and off. We assume that their switching time constants are much faster than the time constants in the fluid domain. Therefore, pumps and pipes with valves are modeled as hybrid systems. In our *GME* paradigm, subsystems are modeled as *components*. Interactions between the components are specified as relations between their *in-ports* and *out-ports*. Components connected by ports define the system model. The ports can model: (i) energy transfer between components in the bond graph framework, and (ii) communication of information by signals. Signals are assumed to have no energy content.

For this work, we build first order linear models[2]. Tanks are modeled as a bond graph segment with a capacitor connected to a 0 junction. Each tank component can have multiple in-ports and out-ports. In-ports have energy connections (bonds) to the 0 junction and out-ports have bonds from the 0 junction. Ports may be marked as in and out based on a conventional direction for energy flow, but this does not mean that energy cannot flow in the opposite direction. In case there is an energy flow in the opposite direction, the corresponding variable takes on a negative value.

Pipes are modeled as resistors connected to a 1 junction. Pipes have exactly one in port and one out-port that can be connected to ports of other tanks and pipes. The switching on the pipes is achieved by specifying switching signals on the 1 junction connected to the resistor. As discussed earlier, pumps are modeled as an effort source connected to a transformer, which is connected to a 0 junction. Pumps have one out-port for representing the pressure delivered by the pump.

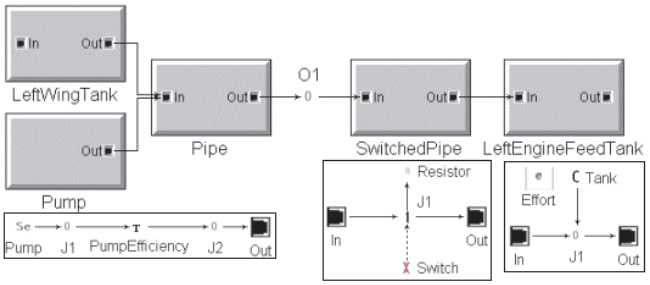

Figure 3: Hierarchical and Compositional Modeling

As an example, the left wing tank is connected to the left feed tank by instantiating two tank components, one pipe component, and one switched pipe component. The switched pipe controls the flow into the feed tank. The out port of the first tank (left wing tank) is connected to the in port of the pipe and out-port of the pipe is connected to the in-port of the second pipe. Since the pump is modeled to pull fuel out of the left wing tank, we connect the out port of a pump component to the in port of the pipe. Fig. 3 illustrates the component based and hierarchical model of this subsystem and the underlying model of the some of the components.

3.4 Modeling for diagnosis

Models form the core component of the tracking and diagnosis algorithms [Biswas and Yu, 1993; Narasimhan *et al.*, 2000]. The hybrid observer uses quantitative *state space models* for tracking nominal system behavior, the fault isolation and identification unit uses temporal causal graphs (*TCG*) for qualitative analysis and input output equation (*IOE*) models for quantitative parameter estimation. We have devised schemes to systematically derive these model representations from the HBG models created in GME.

Precise tracking of nominal system behavior requires the component parameter values in the bond graph model be accurately estimated. We describe our parameter estimation methodology in the next section. For fault isolation and identification, there has to be a a one to one correspondence between faults and parameters in the model. For example, if we abstract a pump model and represent it as an effort source, we cannot differentiate among faults in the electrical versus mechanical/fluid part of the pump. Including a transformer component that models the electrical to fluid energy transformation at an abstract level solves this problem. A partial fault or degradation in the mechanical part of the pump can be attributed to a change in the transformation parameter.

Once the model structure has specified and all parameters have been estimated, the hybrid bond graph model of the complete system is derived by composing the component models and flattening out the hierarchy. The designation of ports as in- and out-ports, and restricting connections to be from out-ports to in-ports only ensures the consistency of bond connections when the components are composed. The

[2]In reality the pressure flow relations are nonlinear.

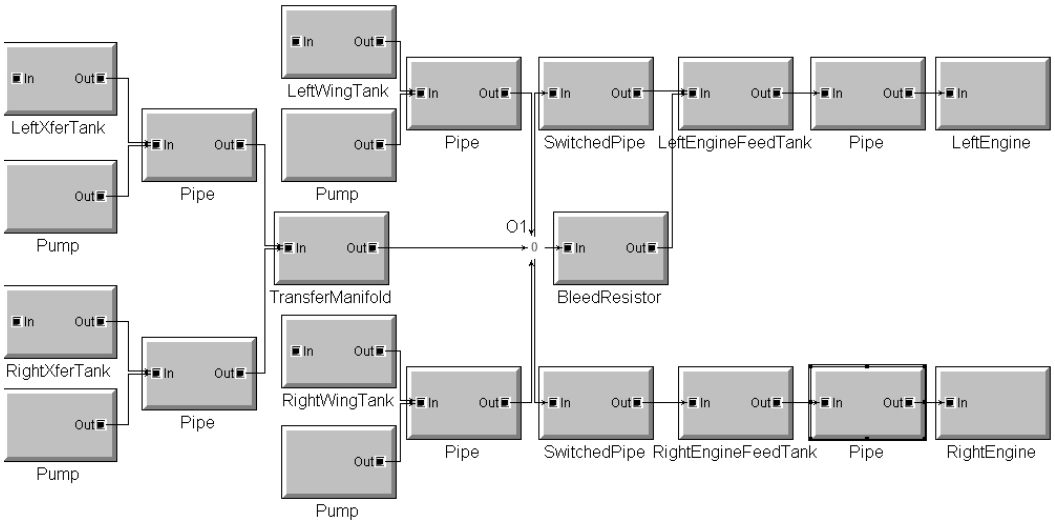

Figure 4: Component Model of Fuel System

resulting hybrid bond graph may be used to systematically derive the state space and temporal causal graph model of the system. In the bond graph framework, each element describes equations that need to be satisfied for that component. For example, for a 0 junction the pressures on all bonds incident is equal and net flow of all bonds is equal to zero. The procedure to convert to state space equations may be summarized as [Karnopp *et al.*, 1990]:

1. Identify state variables (efforts on capacitors and flow on inductors).
2. identify input variables (effort and flow sources).
3. Use constituent equations of the bond graph components to derive the relations between the effort and flow variables in the system.
4. Substitute for all non-state and non-input variables to derive the state equation model. This step is applied repeatedly till all unnecessary variables are eliminated.

The algorithm to derive the TCG from the bond graph is described in [Mosterman and Biswas, 1999].

3.5 Building Models of the Fuel System

From our discussion in earlier sections, the primary model building steps are: (i) identify subsystems and model them at the appropriate level of detail, (ii) compose system models by specifying interactions among the subsystems, and (iii) estimate parameters of the model.

As discussed earlier, tanks, pipes, and pumps are the main components of the fuel system model. In addition, we need to build models for the transfer manifold and the engines. For the scenarios we deal with, it was sufficient to model the engines as constant flow sources, i.e., a sink. Engines have one in-port that represents the flow into the engine from the feed tank. The transfer manifold is modeled as a single capacitor connected to the 0 junction. The transfer manifold has multiple in-ports representing flow into, and multiple out-ports representing flow out of the transfer manifold.

The next step is to determine the complete system configuration. For the fuel system we instantiate 6 tanks: 2 wing tanks, 2 fuselage tanks and 2 engine feed tanks. Each has a corresponding pump. Since the outlets of the wing and fuselage tanks and the inlet of feed tanks all have valves, we created switched pipe components for each of these components. Two instances of the engine are created, and the transfer manifold component is also included in the model. Fig. 4 depicts our component-based GME model of the entire fuel system.

The individual components are connected using bond graph junctions to build the energy model of the overall system. The fuselage tanks supply the transfer manifold, where the flows from the fuselage tanks sum up. This is modeled by connecting one out-port of the fuselage tank to the in-port of a pipe, and the out-port of the pipe to the in-port of the transfer manifold. The pump associated with each tank is also connected to the in port of the pipe. This develops a high pressure at the inlet of the pipe, and hence pulls fuel from the tank into the pipe. The flow from the wing tanks and the transfer manifold combine and distribute evenly to the left and right feed tanks. One out-port of the wing tank is connected to the in port of a pipe. The out-port from these pipes and the out-port from the transfer manifold are connected to a 0 junction to combine the flows. The 0 junction is connected to the in-port of switched pipes whose out-ports are connected to the in-ports of the feed tanks. In order to maintain stability when both feed tanks are closed, a bleed resistor is added to the piping. This resistor bleeds fuel into the left feed tank. The out-ports of the feed tanks are connected to the in-ports of the corresponding engines through pipes.

The next step is to estimate the model parameter values. For the scenarios we model, the engine fuel consumption rate is set at *g gpm* for both engines[3]. All other parameters are estimated from experimental data of an entire fuel burn curve, where all the fuel from the wing and fuselage tanks was con-

[3]In this discussion the actual numbers are not used to avoid any concerns about releasing sensitive data.

sumed by the engines. We used the rate of depletion of fuel in the tanks and the flow out of the tanks when the level control valves are closed to calculate the individual tank capacitances. For the left feed tank the fuel depletion rate is approximately *d lbs/s*, and hence we determine the capacitance of the left feed tank to be $c_l \frac{ft^5 sec^2}{lb}$. Similarly, the right feed tank capacitance is estimated to be $c_r \frac{ft^5 sec^2}{lb}$. We performed similar calculations to determine the wing and fuselage tank capacities (approximately $c_w \frac{ft^5 sec^2}{lb}$). To estimate the resistances, we used the pressure drop and flow through the pipe corresponding to the resistance to calculate the resistance value. The pump effort and efficiency values were given nominal, realistic values.

4 Diagnosis

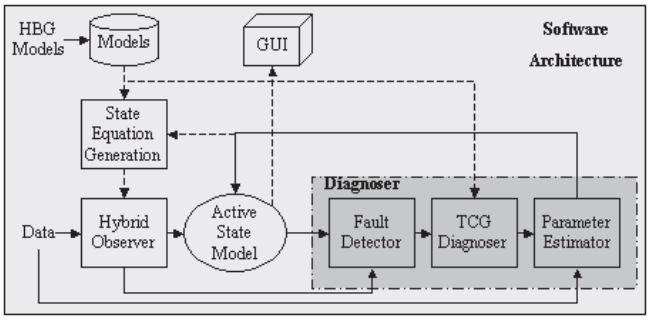

Figure 5: Software Architecture for Diagnosis

The diagnosis task involves tracking dynamic system behavior that includes continuous evolution plus discrete model changes till the fault detector signals a fault. At this point, the fault isolation unit is invoked. Discrete mode changes require dynamic switching of system models, and may also involve discontinuous changes in the system variables. The fault isolation unit also needs to consider change in modes when matching fault signatures with actual system behavior. This motivates the software architecture for diagnosis, illustrated in Fig. 5. The input to the diagnosis system is the model as an *XML* file and the experimental data as a text file. Each line of the data file represents one sample of the data. Although the current version uses a data file as input, replacing it with data from an actual system does not alter the rest of the architecture. Each sample of data includes all input values, all measured output values, and the values of all switching signals. The output of the diagnosis module is the set of diagnostic hypotheses that are consistent with the model and data. The diagnosis output at each time step can be observed in a GUI implemented in Python (*www.python.org*). The active state model (*ASM*) is an internal data structure that maintains information about the system including the current mode, current state estimates, and current diagnostic hypotheses. This structure is updated at each time step from information received from the observer and the diagnosis module. The hybrid bond graph (*HBG*) data structure contains the flattened *HBG* model of the system after composition of all active component subsystems. The *HBG* model also contains the switching conditions for mode changes. These are parsed and stored in the *ASM*. All the diagnosis algorithms modules were implemented in C++. The SWIG toolkit was used for Python-C++ interactions.

The parser reads in the model file, interprets it and creates the *HBG* data structure. The symbolic equation generator (SEG) takes the *HBG* and the current mode of the system and derives the state space equation (*SSE*) model of the system, which is stored in the *ASM*. When tracking system behavior, the hybrid observer reads in the data sample for the next time step from the data file, and checks to see if any controlled (specified in data file) or autonomous (stored in *ASM*) mode changes have occurred. When mode changes occur, the SEG is invoked to re-calculate the *SSE* model. To accommodate for model disturbances and measurement noise, a Kalman filter is built from the current SSE model to track system behavior. At each time step, the fault detector compares the system output with the observer estimates to determine if a fault has occurred in the system. When the fault detector triggers, the diagnosis module is activated. The diagnosis module uses propagation algorithms on the TCG to generate fault candidates that are consistent with the observed discrepancies. Continued tracking by matching the fault signatures generated for each candidate hypotheses helps refine the candidate set. For details on the hybrid observer and diagnosis algorithms, refer to [Mosterman and Biswas, 1999; Narasimhan and Biswas, 2002].

In subsequent sections we briefly describe the hybrid observer and the diagnosis modules and illustrate their functioning by applying them to a diagnosis problem on the fuel system. In the experimental setup, the fuel system is controlled by the sequence defined in Table 1. The data for the experiments was generated using a Matlab/Simulink simulation that was developed at Vanderbilt University. We assume pressure values are measured at the output of each of the six tanks of the fuel system. The fault introduced is an abrupt decrease in the left fuselage tank pump efficiency at time step 200. This occurs in the mode when only the left fuselage tank is supplying fuel, and only the left feed tank is receiving fuel.

4.1 Hybrid Observer and Fault Detector

The hybrid observer tracks the system behavior as it evolves and the fault detector compares the observer output to the system output to determine if a fault is present in the system. The hybrid observer performs the following tasks:

- Continuous tracking of system behavior in current mode,
- Determining if a mode change has occurred, and
- Initializing the observer in a new mode, with the new state and new model.

The discrete time form of the SSE models are derived to track system behavior. To account for model disturbances and noisy measurements, we use a Kalman filter to track system behavior. This requires computation of the R and Q matrices that model the disturbance and noise variances, and K, which

represents the Kalman gain matrix.

$$\dot{\hat{x}} = A\hat{x} + Bu + K(y - \hat{y})$$
$$\hat{y} = C\hat{x}$$
$$\dot{P} = AP + PA^T + BQB^T - KRK^T$$
$$K = PC^T R^{-1}$$

In our experiments, R and Q are diagonal matrices with values of 0.01 along the diagonal. The Kalman gain (K) is initialized to a diagonal matrix of arbitrarily high value (100 in our experiments). This gain matrix typically converges to its true value in a few time steps.

Mode changes may be of two types: controlled or autonomous. Controller issued switching commands need to be provided in the data file. These correspond to the controlled mode changes in the system. At each time step, the observer checks to see if the data set indicates a mode change. All autonomous change conditions are converted so that they contain only state and input variables. The observer uses input data and estimated state values to calculate if the conditions for any autonomous change evaluates to true. This is done at each time step also. For the fuel system, there are no autonomous changes and hence the data file provides sufficient information to determine if a mode change has occurred. If a controlled or autonomous mode change is indicated, the observer computes the new mode. The equation solver is invoked to derive the new *SSE* model. The observer re-initializes the state based on the reset function specified, and continues the tracking in the new mode with a new Kalman filter that is derived from the A and B matrices in the new mode.

Fig. 6 illustrates a sample run of the hybrid observer for the experimental setup described earlier. Gaussian noise with a 2% noise variance was generated using the Matlab models as described earlier. We illustrate the tracking of pressure in the left fuselage tank. The thick line represents the noisy system output (it is more like a waveform than a line due to the noise in the measurements) and the thin line represents the observer estimates. Until time step 200 (at which point the fault was introduced) we notice that this line is completely subsumed by the thick line indicating accurate tracking. However, after time step 200 the thin line deviates from the thick line indicating a fault. The fault detector (uses a 5% detection threshold) flags the fault. In the next section, we describe the fault isolation scheme.

4.2 Fault Isolation and Identification

Our fault isolation and identification methodology, described in greater detail in [Narasimhan and Biswas, 2002], for hybrid systems is broken down into three steps:

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 the parameter of an implicated component whose deviation possibly explains the observed discrepancies in behavior.

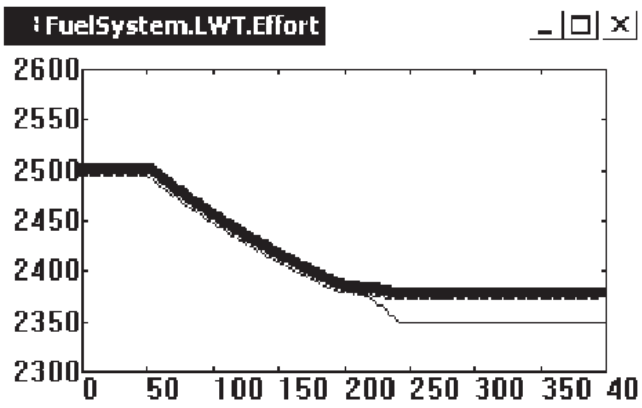

Figure 6: Hybrid Observer Sample Run

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 least squares based estimation techniques 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.

The models used in these three steps, temporal causal graph (*TCG*) and input output equations (*IOE*) model, are derived directly from the hybrid bond graph.

We illustrate the diagnosis algorithms for the experiment discussed in the previous section. As Fig. 5 illustrates, after time step 200 the actual pressure in the left fuselage tank is below the predicted value. The fault detector flags this and triggers the diagnosis process. We use the roll back procedure to propagate this discrepancy back through our models to generate the fault candidates. In the current mode, we get the following candidates: Left Fuselage Pump-, Left Fuselage Pipe+, Transfer Manifold+, Bleed Resistor+, Left Switched Pipe+, Left Feed Pump-. Since the left fuselage tank was not open in any of the previous modes, no candidates are generated in any previous modes.

The next step rolls forward to check for the consistency of the effects of the faults hypothesized against actual system measurements. Since no autonomous mode changes are present and we assume that all controller commands are known, we know exactly what mode the system is in. We generate signatures (effects of fault) in that mode for all the above candidates. In the current mode we cannot distinguish

between the candidates because they have similar signatures. However, when a mode change occurs (right feed tank is also opened), we regenerate signatures in the new mode and note that Left Switched Pipe+ and Left Feed Pump- do not affect the right feed tank pressure. However, we notice a discrepancy in the right feed tank pressure, hence we can drop these candidates. We cannot distinguish between the other candidates (Left Fuselage Pump-, Left Fuselage Pipe+, Transfer Manifold+) with the selected set of measurements. In order to distinguish between these candidates we need more measurements. For example, we could model the pump in more detail and add a sensor to measure the current drawn by the pump motor. This would let us identify faults in the pump as opposed to faults in pipes that the pump is connected to.

Table 2 lists the different fault classes in the fuel system. Each fault class represents multiple instances of the faults in the same component. The fault classes are numbered as follows: 1) Wing Tank Pump (WTP), 2) Wing Tank Pipe Resistance (WTR), 3) Fuselage/Transfer Tank Pump (TTP), 4) Fuselage/Transfer Tank Pipe Resistance (TTR), 5) Transfer Manifold Resistance (TMR), 6) Switched Pipe Resistance (SPR, 7) Feed Tank Pump (FTP), and 8) Feed Tank Pipe Resistance (FTR). The results of our diagnosis experiments for these sets of faults appear in the table. The $\surd$ mark in row i and column j indicates that if the roll back process generated candidates i and j, one of them will be dropped by the roll forward process. The $\times$ mark indicates that the current control sequence and set of measurements are not sufficient to distinguish between the pair in question. In general, the algorithm cannot distinguish between pump and pipe faults associated with the same tank. We need more detailed models and more measurements to do this. We also see that we cannot distinguish between the transfer manifold and fuselage pipe faults. We can distinguish between all other fault classes. The ability to distinguish between all fault classes is critical since the change in control strategy depends on the fault type.

Table 2: Fuel System Fault Diagnosability

	WTP	WTR	TTP	TTR	TMR	SPR	FTP	FTR
WTP	-	$\times$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
WTR	$\times$	-	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
TTP	$\surd$	$\surd$	-	$\times$	$\times$	$\surd$	$\surd$	$\surd$
TTR	$\surd$	$\surd$	$\times$	-	$\times$	$\surd$	$\surd$	$\surd$
TMR	$\surd$	$\surd$	$\times$	$\times$	-	$\surd$	$\surd$	$\surd$
SPR	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	-	$\surd$	$\surd$
FTP	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	-	$\times$
FTR	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\times$	-

5 Conclusions

We have presented a case study on modeling a real system and building a diagnosis engine for the system. The key to successful tracking and diagnosis is to have models that are topologically correct, with parameter value estimates that match the nominal observed behavior so as not to generate false alarms. This can be a difficult and time consuming task, with a lot of experimental data being required to build accurate models. The presence of noise in the data complicates the tracking and fault detection task. For the given set of measurements, our tracking mechanisms worked well with fault-free data provided the variance of the added Gaussian noise was limited to 2%. Part of the reason for such low noise thresholds was the use of a naive threshold-based fault detector in these experiments. The diagnosis system always generated the true fault hypothesis, but in a number of cases the hypothesis set contained more than one fault candidate. This could be attributed to lack of detail in the models and the need for more measurements in the analysis. Also, parameter estimation was not included as part of the experiment. In previous work [Narasimhan and Biswas, 2002], we have shown that parameter estimation often helps to isolate the true fault.

In future work, we would like to build more detailed models of the different components of the fuel system in an attempt to diagnose a larger set of faults. The experiments need to be extended to run with real data provided from Boeing, as opposed to simulated Matlab data that we generated at Vanderbilt University. We would also like to run sensitivity analysis tests to the diagnosis system performance under varying noise and disturbance conditions. Finally we would like to build more robust techniques for fault detection and parameter estimation to combat the effects of noise and disturbance.

6 Acknowledgments

This project was conducted as part of ISIS and the EHS lab at Vanderbilt University. The DARPA/ITO SEC program (F30602-96-2-0227), NASA-IS grant (NCC2-1238), and the Boeing Company, St. Louis have supported the activities described in this paper. We would also like to thank Brian Olson and John Ramirez for helping to build the initial fuel system models.

References

[Biswas and Yu, 1993] Gautam Biswas and Xudong W. Yu. A formal modeling scheme for continuous systems: Focus on diagnosis. In *IJCAI 93*, pages 1474–1479, Chamberey, France, 1993.

[Buck *et al.*, 1994] J.T. Buck, S. Ha, E.A. Lee, and D. G. Messerschmitt. Ptolemy: A framework for simulating and prototyping heterogenous systems. *International Journal of Computer Simulation, Special Issue on "Simulation Software Development"*, 4:155–182, April 1994.

[deKleer and Williams, 1987] Johan deKleer and Brian C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.

[Gertler, 1997] Janos Gertler. Fault detection and isolation using parity relations. *Control Engineering Practice*, 5(5):653–661, 1997.

[Harel, 1987] David Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.

[Hofbaur and Williams, 2002] Michael W. Hofbaur and Brian Williams. Mode estimation of probabilistic hybrid

systems. In *Fifth International Workshop on Hybrid Systems: Computation and Control*, page To Appear, Stanford, California, USA, March 2002.

[Karnopp *et al.*, 1990] D.C. Karnopp, D.L. Margolis, and R.C. Rosenberg. *Systems Dynamics: A Unified Approach*. John Wiley and Sons, New York, 2 edition, 1990.

[Ledeczi *et al.*, 2001] A. Ledeczi, M.Maroti, A. Bakay, G. Karsai, J. Garrett, C. Thomason, G. Nordstorm, J.Sprinkle, and P.Volgyesi. The generic modeling environment. *Workshop on Intelligent Signal Processing*, 2001.

[Lunze, 1999] J. Lunze. A timed discrete-event abstraction of continuous-variable systems. *International Journal of Control*, 72(13):1147–64, 1999.

[McIlraith *et al.*, 2000] Sheila McIlraith, Gautam Biswas, Dan Clancy, and Vineet Gupta. Hybrid systems diagnsois. In *Third International Workshop on Hybrid Systems*, pages 282–295, 2000.

[Mosterman and Biswas, 1998] Pieter J. Mosterman and Gautam Biswas. A theory of discontinuities in physical system models. *Journal of the Franklin Institute : Engineering and Applied Mathematics*, 1(3):401–439, 1998.

[Mosterman and Biswas, 1999] Pieter J. Mosterman and Gautam Biswas. Diagnosis of continuous valued systems in transient operating regions. *IEEE Transactions on Systems, Man, and Cybernetics*, 1(6):554–565, 1999.

[Narasimhan and Biswas, 2001] Sriram Narasimhan and Gautam Biswas. Efficient diagnosis of hybrid systems using model of supervisory controller. In *12th International Workshop on Principles of Diagnosis (DX '01)*, pages 127–134, via Lattea, Italy, March 2001.

[Narasimhan and Biswas, 2002] Sriram Narasimhan and Gautam Biswas. An approach to model-based diagnosis of hybrid systems. In *Fifth International Workshop on Hybrid Systems: Computation and Control*, Stanford , CA, USA, vol. LNCS 2289, pp. 308-322, March 2002.

[Narasimhan *et al.*, 2000] Sriram Narasimhan, Gautam Biswas, Gabor Karsai, Tal Pasternak, and Feng Zhao. Building observers to address fault isolation and control problems in hybrid dynamic systems. In *The 2000 IEEE International Conference on Systems, Man, and Cybernetics (SMC '00')*, pages 2393–2398, Nashville, TN, USA, October 2000.

[Sampath *et al.*, 1996] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D.C. Teneketzis. Failure diagnosis using discrete-event models. *IEEE Transactions on Control Systems Technology*, 4(2):105–124, 1996.

[van Amerongen, 2000] Job van Amerongen. Modeling, simulation and controller design of mechatronic systems with 20sim. *IFAC Mechatronics*, 2000.