An Integrated framework for combining global and local analyses in diagnosing hybrid systems

1Sriram Narasimhan, 2Feng Zhao, 3Gautam Biswas and 4Elmer Hung

1 Dept. of EECS, Vanderbilt University
Nashville, TN 37212
usriram@vuse.vanderbilt.edu

2 Xerox, Palo Alto Research Center
Palo Alto, CA 94304
zhao@parc.xerox.com

3 Dept. of EECS, Vanderbilt University
Nashville, TN 37212
biswas@vuse.vanderbilt.edu

4 Xerox, Palo Alto Research Center
Palo Alto, CA 94304
eshung@parc.xerox.com

Abstract. Sensor-rich systems typically employ extensive signal processing to detect and classify faults for fault isolation tasks. Sensor-poor systems, on the other hand, require models of the system structure and behavior and analytical redundancy techniques to make diagnostic inference. Because of the increasing availability of inexpensive, batch-fabricated micro-controllers and MEMS sensors, it is common for electro-mechanical systems such as office equipment and vehicles to deploy multitude of sensors and microprocessors for control and diagnosis tasks. Most of these systems tend to be embedded systems involving discrete control actions that determine the region of continuous operation of the device. We develop a diagnosis method that combines model-based diagnosis with signal processing techniques to address the challenges in diagnosing complex systems with hybrid discrete/continuous behaviors and to reduce the computational requirements by focusing the signal processing algorithms. We demonstrate the approach on problems in reprographic copier paper path diagnosis, and discuss the computational results.

1. Introduction

Modern systems are complex in nature. They can include supervisory control that switches modes of behavior of the system to optimize system performance. In other words, discrete control actions change the operating region of behavior. The implication of this is that multiple models of the system have to be employed and model switching has to be performed at run time to execute monitoring, fault isolation, and control tasks. The hybrid nature of systems requires new forms of analysis because discrete changes are not handled well by continuous algorithms, and abstracting system behavior to discrete models may result in loss of information critical for fault isolation and control. On the other hand, the availability of cheap sensors makes it possible to obtain a variety of information about the system variables. Most of these sensors provide information about the local functioning of the system that can be used directly to identify system states and isolate faults. The nature of these systems is such that there is a lot of non-local interaction of behavior making it difficult to predict fault characteristics or system states under different
operating conditions. Considering these factors it is important to develop techniques that can handle the hybrid nature of the system and combine global and local analysis.

Model based diagnosis provides a framework for analyzing the global behavior of the system. Given that we have a limited number of sensors, analytic redundancy methods based on model-based techniques have to be applied to derive non-local interaction between the faults and observations.

Signal processing techniques that perform continuous waveform analysis are frequently used to diagnose faults in sensor rich systems where direct correspondence can be established between faults and sensor readings. This provides localized information about individual components in the system. Signal processing is also essential since it is very difficult to build accurate models that incorporate information derived from certain sensors, such as the vibration sensors. The use of such sensors facilitates diagnosis of a richer fault set, and in some cases they maybe the only source of specific kinds of information. Computational complexity may make it infeasible to apply all signal processing algorithms on all signals. Therefore, it is important to develop schemes that allow for selective context driven processing of a signal in an efficient manner. Model based diagnosis enables higher-level reasoning about a more global view of the system, which can be used in conjunction with the localized signal processing techniques. We propose a combination of a model-based approach with signal processing techniques where the model-based diagnosis is able to focus the signal processing to selectively process the available signals.

Current work in model based diagnosis focuses on either discrete event or continuous approaches [SSLS96, L99, MB99c]. Hybrid system diagnosis is usually performed by abstracting the system to continuous or discrete event form. This approach is not sufficient when both continuous behavior and discrete events together are required to generate the necessary diagnostic information. In a hybrid system the behavior evolves continuously until a discrete event causes it to move to a different continuously evolving region in the behavior space. Coming up with continuous representations of hybrid systems can result in very complex non-linear functional relations that are hard to analyze in real time. On the other hand, pure discrete event systems require a lot of simulation and can diagnose only qualitative faults. We propose a diagnosis methodology that uses hybrid models of the system and thereby performs hybrid diagnosis.

We have implemented a prototype system that supports a set of the functionalities presented in this paper.

2. An Example System

We motivate the need for hybrid model based diagnosis, signal processing and an effective way to combine the two through an example. We look at a paper moving system (Figure 1) that forms a sub system of reprographic machines. Such a system consists of a paper tray with sheets of paper, a paper path to a destination point, and rollers along the paper path that move the sheet of paper forward. Motors drive the rollers through gear assemblies. Rollers may be stationary or mobile in the vertical direction. Typically mobile rollers are placed above the paper stack, and are dropped onto the stack by energizing solenoids connected to them. Once in contact, the roller moves a sheet forward. When this sheet has moved forward.
to the next roller, the mobile roller is retracted (by de-energizing the corresponding solenoid) to prevent the next sheet from moving forward till its allocated time.

This system behavior consists of discrete events like the switching on and off of motors, and energizing and de-energizing of solenoids. The paper motion, currents drawn by motors and solenoids and vibration signals produced are all examples of continuous behaviors. Current system configurations include a paper presence sensor at some point along the paper path to detect the arrival of the leading and trailing edge (relative to leading edge) of the sheet when the output changes from 0 to 1, and vice versa. Accelerometers are also strategically placed to pick up vibration signals from the motors and solenoids. These are the only sensors available in the current test bed. Useful knowledge about the health of the components can be obtained by analyzing continuous waveform data from the different types of sensors. Vibration sensors are a good example of this. They are generic, can capture data about different electro-mechanical components without intimate knowledge of the system configuration and wiring, and are representative of the type of high bandwidth data that can be used for health monitoring and fault isolation in a distributed setting.

We use models to simulate the nominal behavior of the system. If the paper sensor and predictions from models indicate that leading edge of the sheet arrived later than expected, this could be due to a variety of reasons such as motors running slow, solenoid energizing slowly, and motors not ramping up to speed as fast as expected. We can use the model to identify the possible causes of this deviation. We continue to observe the trailing edge of the sheet. If it is on time (after correcting for the delay caused by the late arrival of the leading edge), then we can eliminate the motor running slow hypothesis because the models would predict the trailing edge to be late if the motor is running slow. The level of detail in our models and the current set of sensor readings make it very difficult to distinguish between the two remaining fault hypotheses using a model-based approach. Therefore, we switch to an analysis of the vibration signals recorded from the accelerometers. The presence or absence of certain signal characteristics tells us whether the energizing process was normal or abnormal. If it turns out to be normal, we can eliminate the solenoid and uniquely identify the fault.

3. Framework for Hybrid diagnosis

In our proposed diagnosis methodology (Figure 2), we build a hybrid observer using the constraint equations of the system. This enables us to track nominal behavior by assigning nominal values to the parameters in the equations. If the predictions differ from the measurements we start the fault isolation process. An initial candidate set is identified by back propagating the discrepancy through the temporal causal graph of the system. The effects of the candidate faults on future behavior can be determined by forward propagating through the temporal causal graph to generate qualitative fault signatures. These signatures can be compared to measurements to refine the candidate set. We try to estimate the deviated parameters for the remaining candidates and build fault observers using these estimated parameters values. The quantitative predictions can be used to further refine the set. Depending on the make up of the candidate set, we can perform selective signal processing to eliminate some of the candidates.

![Figure 2: Overall diagnosis methodology](image)
We make the following assumptions in our work.

1. Only single faults occur.
2. All faults are associated with parameters that deviate when the fault occurs.
3. We have access to the control signals. (We do not assume a specific control sequence though). Control signals are used to determine mode transitions.

In the next three sections we will discuss in more detail our modeling paradigm, the fault detection algorithm, and the fault isolation algorithm.

3.1 Modeling for diagnosis

Parsimonious and effective models form the key to developing efficient diagnosis algorithms. The model based diagnosis methodology is only as good as the models it is based on. The type of models and the level of modeling detail vary depending the problem at hand and the required precision in analysis. However, building system models that provide all the information necessary to identify faults in the system is essential for the model based diagnosis methodology to be useful. Since we perform both qualitative and quantitative analysis we would require two different kinds of models.

Earlier attempts at modeling tried to abstract hybrid systems as either discrete event systems or continuous systems [SSLSD96, L99]. Traditionally, discrete event systems are modeled by finite state automata, directed graphs and petri nets whereas continuous systems are typically represented by differential equation models, block diagrams and signal analysis graphs. Although it is true that most hybrid systems are continuous systems at the lowest level of detail, building hybrid models proves useful in simulating the behavior of the system. It seems natural to capture the hybrid nature of the system in models.

3.1.1 Hybrid constraint models

The analytic models that define hybrid behavior should facilitate the building of nominal and faulty observers. In addition, we would like to use the models for parameter estimation. The analytic hybrid models of such systems consist of three parts

i) Modes the system can be in
ii) Constraints that hold within each mode (Ordinary differential & algebraic equations)
iii) Transitions between modes (Constraints defined in terms of discrete events)

One of our goals when building the models is to make the models compositional. We would like to be able to build models of different devices independently (a generic model library can be built), instantiate them for particular problems, and connect these devices together to compose the system model for specific tasks. Hence we specify the hybrid models of individual components and the connections between them to synthesize the system model in terms of the three parts mentioned above.

HCC (Hybrid concurrent constraint language) [GJSB95] is a modeling language that facilitates building of hybrid models. It includes constructs that can be used to specify rules of transition and constraints on the variables. In addition it also provides tools to build compositional models. Last, but not least, it provides a built in simulator to generate system behavior given the initial conditions. These features make HCC a powerful hybrid modeling language and hence the language of our choice.

Generic component models

The component models are developed in a generic fashion. This facilitates building a library of component models that can be used to build any system model that includes the component. Also, one generic model of a component can be used to instantiate any number of occurrences of the same component in the system.
In order to make the models generic we have to make certain information available only at run time. These include inputs, outputs, control signals, and initial values of variables and parameters of the system. Occurrences of the component can be instantiated by assigning appropriate values to these arguments.

The component models consist of the same three parts as the system model i.e. modes, constraints in each mode and transition function between modes.

The modes of a component depend on the physical properties of the component. For example, a motor could be OFF, SPEEDING_UP or RUNNING_AT_MAX_SPEED. The number of modes would also dependent on the purpose of the models. For certain applications the SPEEDING_UP mode of the motor may not be of any significance, whereas for other applications the SPEEDING_UP mode may affect other parameters and variables in the system significantly to require explicit modeling.

Once the modes of the components have been identified, we can specify the constraints that govern each of these modes. These can be specified as relations on the variables and parameters of the system. They can be built from system equations obtained from handbooks, manuals or domain experts. These constraints can easily derived from the ODE’s of the system. For systems that are not expressed as ODE’s these constraints can be derived by looking at the structure and behavior of the system specified by experts or available from operating manuals. The constraints to be used in any mode would depend on level of detail required which would in turn be dependent on the application and information available. Examples of constraints are,

\[
\begin{align*}
\text{velocity_of_sheet} & = k \ast \text{speed_of_roller} \\
\frac{\partial}{\partial t} (\text{leading_edge_of_sheet}) & = \text{velocity_of_sheet}.
\end{align*}
\]

The third component of the models is the transition function that specifies the rules that trigger transitions from one mode to another. There are two possible causes for transitions. Control signals (external) cause the component to change modes. For example, when the motor is in the OFF mode and it receives an ON signal, it transitions to the SPEEDING_UP mode. Autonomous jumps (internal) may also cause mode transitions. Autonomous jumps are caused when variables in the system reach certain landmark values. For example, all motors have a maximum torque that they can generate. No matter how much power is supplied, the motor cannot generate any more torque. In the SPEEDING_UP mode when the motor torque reaches it’s maximum value, the motor transitions to the RUNNING_AT_MAX_SPEED mode where the set of constraints are different from the SPEEDING_UP mode.

Figure 3 shows the HCC code for a part of the motor model.

```hcc
// OFF mode - Maintain torque at 0 until ON signal is received
SteadyZero = ()
    do always torque' = 0 watching ON,
    when ON do Increasing()

// RAMPING_UP mode
INCREASING – The hcc code looks like this
Increasing = ()
    do always torque = k_rup_time * In.value watching (OFF || torque = k_max_speed * maximum_torque),
    when (OFF || torque = k_max_speed * maximum_torque) do {
        if OFF then Decreasing(),
        if torque = k_max_speed * maximum_torque then unless OFF then SteadyMax()
    }

// ON mode - Motor torque is maintained at max value
SteadyMax = ()
    do always torque' = 0 watching OFF,
    when OFF do Decreasing()

// RAMPING_DOWN mode
Decreasing = ()
    do always torque' = k_ramp_down watching (ON || torque = 0),
    when (ON || torque = 0) do {
        if ON then Increasing(),
        if torque = 0 then unless ON then SteadyZero()
    }
```

This example illustrates four modes of a motor namely, OFF, RAMPING UP, ON and RAMPING DOWN. The constraints are specified within each mode. For example in the RAMPING DOWN mode the torque steadily reduces to 0, which is achieved by setting the derivative of torque (torque’) to a small constant. The transitions from this mode are triggered either by a MOTOR ON control signal (transitions to RAMPING UP mode) or the torque reaching 0 (transitions to OFF mode).

**Composing system model for generic component models**

Given the individual component models and the structure of the system, the system model can be automatically synthesized by aligning the corresponding input and output connections of the components. The total number of modes in the system is a cross product of the number of modes of the individual components. The constraints within each mode of the system would be derived from the constraints in each of the individual components. The transition function still remains the same. The point to note is that because of the input/output interaction of the components, changes in one component may affect the variables in another component thus causing a sequence (more than one) of mode transitions at a particular instant in time.

For purposes of composing the system model we introduce the concept of a generic connection box that plays the role of a placeholder to transfer values between various components. The Generic connection box consists of a single variable. This single variable takes on the value obtained at the input port and passes on this value to the output port. The connection box can also be used as a transfer function in which case the value at the output port is determined by the value at the input port and the transfer function. All the transfer functions in our work are 1.

The HCC model of the connection box looks like:
```hcc
always Connection = () [value] {  
  always temp = 0  
},
```

Now the structure of the system can be specified through connection boxes. For example if component A is connected to component B then we can create a connection box A_B whose input port is connected to the output of component A and whose output port is connected to the input of component B. The system structure is specified through a number of such connection boxes by making associations between input and output ports. The system behavior model is automatically synthesized from the structure and individual component behavior models.

The structure of the example system, illustrated in Figure 1, can be specified in Figure 4.

---

**Figure 3: Model of motor in HCC**

**Figure 4: Composing system model in HCC**
### 3.1.2 Qualitative temporal causal graph models

Temporal Causal Graphs (TCG) are directed graphs that capture system dynamics in the form of algebraic and temporal constraints among the variables in the system. The nodes in the graph represent the variables in the system and directed edges between them capture the cause-effect relation between them. For example, an edge directed from node A to node B indicates that a change in the variable associated with node A would cause a change in the variable associated with node B. This relation could be algebraic (defined by proportionalities and equalities) or temporal (integral). In case of an algebraic relation the effect is immediate whereas in case of a temporal relation there is a time delay in the effect. Some edges are also labeled with component parameters. This indicates that the parameter is a component of the functional relation between the two variables. If an edge between nodes A and B exists with the label parameter P and we see that B has deviated from normal, then it could either have been because A deviated or because P has deviated. This property allows us to perform back propagation in the graph to identify fault candidates given deviated measurements and forward propagation to identify the future effects of the hypothesized faults.

![Figure 5: TCG for paper path](image)

The TCG of the paper path is illustrated in Figure 5. For example the edge between X_le and T_le indicates that X_le affects T_le and the –1 on the edge indicates that it is an inverse relationship. In other words if the X_le is – (position of sheet is behind where it should be) then T_le will be + (it will arrive late at the sensor location. The relation between v_sheet and X_le is integral, i.e. there is time delay in the effect of change in v_sheet on X_le. The edge between X_le and X_te is governed by a parameter k_size. The k_size parameter is directly linked to the choice of paper tray that has been selected. A value of + (-) for k_size implies that the paper size selected is larger (smaller) than normal. Hence if X_le is as expected and the correct paper size has been selected (k_size = 0) then X_te is also as expected whereas if a larger than the expected paper size has been selected (k_size = +) then X_le is –.

### 3.2 Fault Detection

The fault detection task identifies if the system behavior is deviating from the nominal behavior, and if that is the case an initial set of fault hypotheses that can explain the deviation. The fault detection process is illustrated in figure 6. The three main steps in fault detection are

1) simulating system behavior
ii) using an observer to map discrepancies in predictions and observations to noise, a mode change or a fault, and
iii) identifying an initial candidate fault set and generating the prediction signature for the candidates in fault set

**Simulator** – As mentioned earlier, HCC provides a built in simulator that allows us to track nominal system behavior. We initialize the HCC models by specifying the parameter values and initial values under normal conditions. For example,

Motor(FM,Power_to_feed_motor,Feed_motor_output_torque,0,k_rup_time_fm,k_max_speed_fm,15.21,F M_ON,F M_OFF),

specifies that the initial value of the speed is 0 and the parameters associated with the motor (max speed and ramping up) are 1 and 5.21 respectively. Once all the components are initialized, the system behavior is automatically simulated (by HCC’s built in simulator) based on controller events. This provides quantitative predictions for the system variables as a function of time.

**Observer** – The observer has to compare the predictions from the simulator with the actual observations from the system. We assume the presence of a simple median filter and use heuristics such as an observed discrepancy persists for a few time steps before it is reported. We assume that this rather simplistic technique is sufficient to cut down on the noise and avoid false alarms. Reported discrepancies in measurements can now be attributed to mode changes in the system or due to occurrence of a fault. If the discrepancy indicates a mode change then the simulator has to be informed of this change and appropriately restarted in the right mode with the correct initial conditions. In case of a fault condition, an initial candidate set has to be identified and the fault isolation task is initiated.

In our work, we assume that all mode transitions can be determined outside the observer. This requires that we have access to the control signal and our models are good enough to identify autonomous jumps. We also use a simple threshold value to distinguish between noise and a faulty condition. In other words if the discrepancy between predictions and observations are within a threshold, it is attributed to noisy measurements and if the discrepancy is above the threshold then it is attributed to a fault condition.

![Figure 6 : Fault Detection](image)
**Discrepancy Detection** - The observer compares the predictions from the models with the corresponding sensor readings to check for discrepancies. In our work, we say that there is a discrepancy if the difference between predictions and observations is greater than a threshold (defined as a percentage change from nominal value). More sophisticated and complicated algorithms may also be used for discrepancy detection.

**Initial Fault hypotheses and generating signature** – Once a fault condition has been detected, we use the temporal causal graph models to identify an initial set of fault candidates. The discrepancy in qualitative form is back propagated through the tcg to identify the possible causes for this discrepancy. The candidate would include the faulty component and the mode in which the fault could have occurred. The back propagation has to be done across past mode transitions since a fault that occurred in an earlier mode may manifest in a later mode. This is essential due to the hybrid nature of the system. The qualitative value of the discrepancy is back propagated through the causal graph to identify an initial candidate set. In back propagation we traverse the graph against the arrows and if an edge with a parameter label is traversed, that parameter is included in the candidate set.

For example, if we see that T_le is + (leading edge of paper reaches sensor late) in Figure 3 we can propagate backwards in the current mode to indicate that (w_roller -) might be responsible or (T_pullin_solenoid -) might be responsible. Now w_roller could be - because t_motor was - or k_speed was -. Hence we get a candidate k_speed -.

For each of the candidates in the candidate set, we can qualitatively predict future behavior by forward propagating the effects of that fault (deviated parameter) through the temporal causal graph to get the signature of the fault. The signatures are in the form of above, same as or below nominal behavior values for magnitude and higher order derivatives for the variables in the system.

For example, if k_speed – is a candidate, we can forward propagate the effects to w_roller – and so on to x_le – (Figure 3). We can further propagate to X_te – and T_te + (trailing edge of paper arrives at sensor late relative to the arrival time of the leading edge). So our prediction would be that the trailing edge should also be late in getting to the sensor.

For more details about the back propagation and forward propagation algorithms refer to [MB99c].

**3.3 Fault Isolation**

Figure 5 illustrates our fault isolation methodology.

---

**Figure 7: Fault Isolation**

There are three ways in which we can refine the candidate set. Any one or more of these steps can be applied at any time to eliminate candidates from the set.
**Qualitative analysis through progressive monitoring** – We already have qualitative signatures (predictions) for each of the candidates in the fault set. These can be compared to the sensor readings that have been converted to qualitative form, i.e., above, equal, or below the expected magnitude. Any candidate whose qualitative prediction and observation do not match can be dropped from the set. This comparison is carried out over time and is based on the principle of progressive monitoring. Higher order effects slowly manifest themselves in the magnitude and slope of the signal. Hence we only measure the magnitude and slope of a signal but try to reconcile them with the signature by moving higher order predictions down as time progresses. For more details on progressive monitoring refer to [MB99c].

**Quantitative simulation using constraint models** – We can overcome some of the limitations of qualitative analysis, by estimating the deviated parameter value for each candidate. Once this parameter has been estimated it can be plugged into the constraint models to simulate the behavior of the system under the hypothesized fault condition. This simulation would take into account mode switches. If the predictions from the simulation do not match actual sensor readings, the corresponding candidate can be dropped from the set. As in the case of discrepancy detection, a simple threshold is used to identify mismatches. For example, if we are somehow able to measure the velocity of the sheet and the speed of the roller, we can easily estimate the parameter k using the relation between the measured variables.

Unobservable parameters, lack of accuracy and detail in models and noise in sensor readings make the parameter estimation task difficult. Moreover, if the model is complex and non linear, online estimation by numerical methods is difficult because of stiffness and convergence problems. Therefore, signal processing techniques play an important role in diagnosis.

**Signal processing for diagnosis** – Depending on the make up of the candidate set we run specific signal processing algorithms (after choosing relevant time windows) and based on the results of the test we can decide to drop the candidate or not. For example, we can run a principal component analysis on the vibration signal when the solenoid is being energized to check if it energized normally or abnormally.

![Signal processing for diagnosis](image)

**Figure 8 : Signal processing for diagnosis**

Signal processing techniques are now being widely used for fault diagnosis. In its simplest form, test data is used to learn a classifier and actual data is then sent through the classifier to identify a health index. The health index would indicate whether a particular actuator is performing a specific operation normally or in one of its fault modes. In our work, we use simple ranges of values of the health index to identify the mode but a more complicated function of the health index can also be used.

We have developed a set of adaptive signature analysis algorithms for analyzing distributed vibration data and have demonstrated the algorithms on a reprographic copier paper drive plate test bed comprising multiple motors and solenoids. The diagnosis of component faults and conditions using a set of distributed vibration sensors poses several challenges. In-situ monitoring and diagnosis require real-time processing of high-bandwidth vibration data. Robust diagnosis requires pre-processing the data that are often very noisy and exploiting information from all the available sources. Our approach has three main components:
Signal processing using Wavelet and STFT to extract signal features indicative of component degradation and faults; (2) Compressing high-bandwidth data by Principal Component Analysis; (3) Fusing multiple sensor data streams using a Bayesian decision analysis in a composite feature space. The algorithm has been successfully applied to classifying motor and solenoid faults on the copier test bed. Because the algorithm attains its adaptivity through online training on lifetime test data, we believe it also applies to many other applications that require distributed sensor data analysis.

Sample run

We present below an example of how the integrated system is employed to derive the true fault associated with the system. LE (TE) indicates the arrival time of the Leading Edge (Trailing Edge) of the sheet at the paper sensor position (Figure 1). Hence LE 0 (+,-) indicates that the leading edge was on time (early, late). Vib_pullin indicates the result of applying the principal component analysis on the vibration signal form the plate when the solenoid was pulling in. Vib_pullin 0 (1) indicates that pull in was normal (abnormal).

Observation 1: Leading edge of sheet late at wait station sensor --> LE +
Faults consistent with Observation 1
  • Solenoid pull in time high
  • Motor ramping up time high
  • Motor nominal speed not reached

Observation 2: Trailing edge of sheet on time at wait station sensor --> TE 0
Faults consistent with observations 1 and 2
  • Solenoid pull in time high
  • Motor ramping up time high

Observation 3: Vibration data indicates Solenoid pull in time is ok --> Vib_pullin 0
Faults consistent with Observations 1, 2 and 3
  • Motor ramping up time high

Discussions and Future Work

We develop a diagnosis methodology that combines a model-based approach with signal processing in an efficient way. It is a hybrid system diagnosis methodology that uses hybrid models of the system. The modeling scheme is compositional and scalable. Changes in structure of the system can be incorporated very easily into the models. We have implemented a prototype system for a reprographic device that supports a set of the functionalities presented above.

This work tries to bridge the gap between purely discrete event and continuous system diagnosis. We do not require the extensive simulation (for pre compilation) some of the discrete event systems require [L99 and SSLSD96]. On the other hand we reduce some of the computational complexity of continuous systems by eliminating candidates based on qualitative information only. We also provide a framework for integrating model-based diagnosis that performs global analysis and signal processing that performs localized analysis.

In our future work we would like to build more generic observers that can determine if a control signal was issued and accordingly make the mode transitions. We are also trying to develop filters that can distinguish between noise and deviations due to a fault. We are working on building robust parameter estimation techniques. We would also like to build a database of signal processing techniques that provide different kinds of information. Currently we are trying to build a diagnosis engine for the three-tank system based on the framework presented.

Acknowledgement. This work was conducted at Xerox Palo Alto Research Center as part of the Smart Matter Diagnostics Project and at Vanderbilt University as part of the Modeling and Analysis of Complex
Systems Lab. We thank Markus Fromherz, John Everett, Meera Sampath, and Peter Struss for helpful discussions and feedback on the work, and Vineet Gupta for help with the HCC programming language.

References:


