Practical Implementation of Diagnosis Systems Using Timed Failure Propagation Graph Models

Sherif Abdelwahed, *Senior Member, IEEE*, Gabor Karsai, Nagabhushan Mahadevan, and Stanley C. Ofsthun

***Abstract*—Timed failure propagation graphs (TFPGs) are causal models that capture the temporal aspects of failure propagation in typical engineering systems. In this paper, we present several practical modeling and reasoning considerations that have been addressed based on experience with complex real-time vehicle subsystems. These include handling intermittent faults, reasoning over dynamically commanded test sequences, dealing with the constraints of limited computational resources, and providing automated model verification. We finally present a vehicle subsystem case study.**

***Index Terms*—Consistency-based reasoning, diagnosis system implementation, intermittent fault, model-based diagnosis, test alarms.**

I. Introduction

THE ABILITY to rapidly react to unpredictable and dramatic changes in both the system and its environment is a key requirement for future engineering systems and applications. The successful operation of these systems relies on the existence of efficient mechanisms to detect, isolate, and correct problems early and to reallocate resources where they are most needed. Model-based diagnosis techniques [1] help address these issues by enabling run-time analysis and providing effective means to identify operational problems at early stages.

In an earlier work [2], a model-based diagnosis approach was developed for a general class of engineering systems based on the timed failure propagation graph (TFPG) model. TFPGs [3], [4] are causal models that describe the system behavior in the presence of faults. The TFPG structure captures the effect of timing constraints and switching dynamics on the propagation of failures in practical discrete-event and hybrid systems. A TFPG modeling and reasoning tool has successfully been developed and used in an integrated fault diagnoses and process control system [5].

The temporal aspects of the TFPG model are closely related to the domain-theoretic notion of temporal dependency proposed in [6]. However, there are several major differences between the two approaches. In particular, a TFPG-based diagnosis implements a real-time incremental reasoning approach that can handle multiple failures including sensor/alarm faults. In addition, the underlying TFPG model can represent a general form of temporal and logical dependency that directly incorporates the dynamics of multimode systems.

In [7], the authors presented a consistency-based approach for the robust diagnosis of systems in which failure behavior can be captured by TFPG models. TFPG diagnosis is conducted at run time on an incremental nonmonotonic reasoning approach that is robust with respect to sensor faults. The underlying algorithm implements two main procedures. The first procedure generates an optimal consistent initial state assignment based on current state observation. The other procedure generates the set of all consistent hypotheses from a given initial state assignment. The hypotheses are then evaluated based on a given plausibility measure. The most plausible hypotheses are then presented in the output failure report.

In this paper, we present several practical modeling and reasoning considerations that have been addressed within the TFPG diagnosis system based on experience with practical real-time vehicle subsystems [8]. In particular, we address intermittent faults, dynamic test sequences (test alarms), limited computational resources, and automated model verification. We discuss the practical context of these problems and introduce the modeling and reasoning extensions that have been developed to handle these issues. Although these issues have mainly been tackled for vehicle subsystems, the underlying extensions to the TFPG diagnosis system are applicable to a wide variety of practical engineering systems.

The problem of intermittent faults has been addressed in the literature for various diagnosis models. In [9], an extension to the conventional consistency-based diagnosis algorithms is introduced to handle intermittent faults. The work in [10] converts intermittent diagnosis tasks of combinational logic to dynamic programming. In [11], an intermittent diagnosis approach is presented for a general class of discrete-event systems that assume intermittent faults are followed by a reset event. This assumption is relaxed in [12] for a single intermittent fault or spurious alarm. The diagnosis of intermittent failure for an industrial process modeled as Petri nets is introduced in [13]. To the best of our knowledge, there is no current literature on handling intermittent faults or test alarms for diagnosis systems based on causal models.

This paper is organized as follows. In Section II, the TFPG model is introduced. Section III presents an overview of the consistency-based diagnosis approach for TFPG models. Section IV introduces the extensions to the TFPG model and its

Manuscript received December 18, 2007; revised April 28, 2008. First published October 10, 2008; current version published January 5, 2009. The Associate Editor coordinating the review process for this paper was Dr. John Sheppard.

S. Abdelwahed is with the Department of Electrical and Computer Engineering, Mississippi State University, Mississippi State, MS 39762 USA (e-mail: sherif@ece.msstae.edu).

G. Karsai and N. Mahadevan are with the Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN 37235 USA.

S. C. Ofsthun is with Boeing Integrated Defense Systems, St. Louis, MO 63166 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIM.2008.2005958

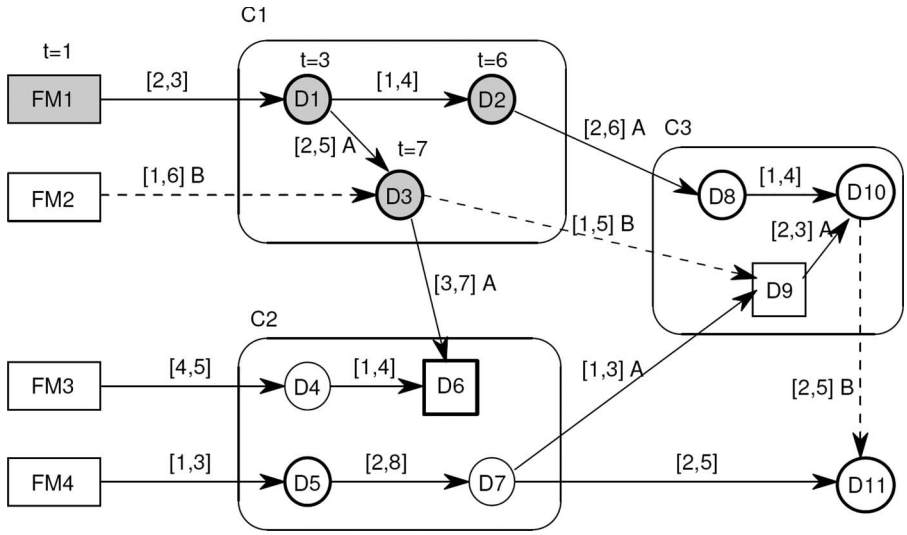

Fig. 1. TFPG model ($t = 10, \text{Mode} = A \forall t \in [0, 10]$).

reasoning algorithm to handle intermittent faults, test alarms, model verification, and computational resource limitations. A vehicle subsystem case study is discussed in Section V. Conclusion and future works are presented in Section VI.

II. TFPGs

A TFPG is a labeled directed graph where nodes represent either failure modes (which are fault causes) or discrepancies (which are off-nominal conditions that are the effects of failure modes). The edges between nodes in the graph capture the effect of failure propagation over time in the underlying dynamic system. To represent failure propagation in multimode (switching) systems, edges in the graph model can be activated or deactivated depending on a set of possible operation modes of the system. Formally, a TFPG is represented as a tuple $(F, D, E, M, \mathsf{ET}, \mathsf{EM}, \mathsf{DC})$, where we have the following.

F	Nonempty set of failure nodes.
D	Nonempty set of discrepancy nodes.
$E \subseteq V \times V$	Set of edges connecting the set of all nodes $V = F \cup D$.
M	Nonempty set of system modes. At each time instance t, the system can be in only one mode.
$\mathsf{ET} : E \to I$	Map that associates every edge in E with a time interval $[t_1, t_2] \in I$.
$\mathsf{EM} : E \to \mathcal{P}(M)$	Map that associates every edge in E with a set of modes in M. We assume that $\mathsf{EM}(e) \neq \varnothing$ for any edge $e \in E$.
$\mathtt{DC} : D \to \{\mathtt{AND}, \mathtt{OR}\}$	Map defining the class of each discrepancy as either AND or OR node.
$\mathtt{DS} : D \to \{\mathtt{A}, \mathtt{I}\}$	Map defining the monitoring status of the discrepancy as either A for the case when the discrepancy is active (monitored by an online alarm) or I for the case when the discrepancy is inactive (not monitored).[1]

[1]In this paper, we will interchangeably use the terms alarms and monitored discrepancies as they mean the same thing.

In the preceding model, the map ET associates each edge $e \in E$ with the minimum and maximum times for the failure to propagate along the edge. For an edge $e \in E$, we will use the notations $e.\text{tmin}$ and $e.\text{tmax}$ to indicate the corresponding minimum and maximum times for failure propagation along e, respectively. That is, given that a propagation edge is enabled (active), it will take at least (most) tmin (tmax) time for the fault to propagate from the source node to the destination node. The map EM associates each edge $e \in E$ with a subset of the system modes at which the failure can propagate along the edge. Consequently, the propagation link e is enabled (active) in a mode $m \in M$ if and only if $m \in \mathtt{EM}(e)$. The map DC defines the type of a given discrepancy as either AND or OR. An OR-type discrepancy node will be activated when the failure propagates to the node from any of its parents. On the other hand, an AND discrepancy node can only be activated if the failure propagates to the node from all its parents. We assume that TFPG models do not contain self loops and that failure modes are always root nodes, i.e., they cannot be a destination of any edge. Also, a discrepancy cannot be a root node, that is, every discrepancy must be a successor of another discrepancy or a failure mode.

Fig. 1 shows a graphical depiction of a failure propagation graph model. The rectangles in the graph model represent failure modes, whereas the circles and squares represent OR- and AND-type discrepancies, respectively. The arrows between the nodes represent failure propagation. Propagation edges are parameterized with the corresponding interval $[e.\text{tmin}, e.\text{tmax}]$ and the set of modes at which the edge is active. Fig. 1 also shows a sequence of active discrepancies (alarm signals) identified by shaded discrepancies. The time at which the alarm is observed is shown above the corresponding discrepancy. Dashed lines are used to distinguish inactive propagation links.

The TFPG model captures observable failure propagations between discrepancies in dynamic systems. In this model, alarms capture state deviations from nominal values. The set of all observed deviations corresponds to the monitored discrepancy set in the TFPG model. Propagation edges, on the other hand, correspond to causality (for example, as defined by energy flow) in the system dynamics. Due to the dynamic nature

of the system, failure effects take time to propagate between the system components. Such time, in general, depends on the system's time constants as well as the size and timing of the underlying failure. Propagation delay intervals can be computed analytically or through simulation of an accurate physical model.

Failure propagation in a TFPG system has simple semantics. The state of a node indicates whether the failure effects reached this node. For an OR-type node v' and an edge $e = (v, v') \in E$, once a failure effect reaches v at time t, it will reach v' at a time t', where $e.\text{tmin} \leq t' - t \leq e.\text{tmax}$. On the other hand, the activation period of an AND alarm v' is the composition of the activation periods for each link $(v, v') \in E$. For a failure to propagate through an edge $e = (v, v')$, the edge should be active throughout the propagation, that is, from the time the failure reaches v to the time it reaches v'. An edge e is active if and only if the current operation mode of the system m_c is in the set of activation modes of the edge, that is, $m_c \in \text{EM}(e)$. When a failure propagates to a monitored node $v'(\text{DS}(v') = \text{A})$, its physical state is considered ON; otherwise, it is OFF. If the link is deactivated any time during the propagation (because of mode switching), the propagation stops. Links are assumed memoryless with respect to failure propagation so that the current failure propagation is independent of any (incomplete) previous propagation. Also, once a failure effect reaches a node, its state will permanently change and will not be affected by any future failure propagation. Note that this original assumption does not accommodate reasoning about intermittent faults. This will be relaxed later in this paper.

III. Consistency-Based Reasoning Approach

The reasoning algorithm for TFPG model diagnosis is based on a consistency relationship defined using three state mappings for the graph nodes of the TFPG model: physical, observed, and hypothetical.

A *physical system* state corresponds to the current state of all nodes in the TFPG model. At any time t, the physical state is given by a map $AS_t : V \to \{\text{ON}, \text{OFF}\} \times \mathbb{R}$, where V is the set of nodes in the TFPG model. An ON state for a node indicates that the failure (effect) reached this node; otherwise, it is set to OFF. The physical state at time t is denoted $AS_t(v).\text{state}$, whereas $AS_t(v).\text{time}$ denotes the last time at which the state of v is changed. Failure effects are assumed permanent; therefore, the state of a node once changed will remain constant after that. A similar map is used to define the state of edges based on the current mode of the system.

The *observed state* at time t is defined as a map $S_t : D \to \{\text{ON}, \text{OFF}\} \times \mathbb{R}$. Clearly, the observed states are only defined for discrepancies. The observed state of the system may not be consistent with the failure propagation graph model temporal constraints due to potential alarm failures. However, we assume that the monitored discrepancy signals are permanent so that once the observed state of a discrepancy is changed, it will remain constant after that.

The aim of the diagnosis-reasoning process is to find a consistent and plausible explanation of the current system state based on the observed state. Such an explanation is given in the form of a valid hypothetical state. A *hypothetical state* is a map that defines node states and the interval at which each node changes its state. Formally, a hypothetical state at time t is a map $H_t^{V'} : V' \to \{\text{ON}, \text{OFF}\} \times \mathbb{R} \times \mathbb{R}$, where $V' \subseteq V$. Similar to actual states, hypothetical states are defined for both discrepancies and failure modes. The estimated earliest (latest) time of state change is denoted $H(v).\text{terl}$ ($H(v).\text{tlat}$).

A hypothetical state is an estimation of the current state of all nodes in the system and the time period at which each node changed its states. An estimation of the current state is valid only if it is consistent with the TFPG model. The state consistency in TFPG models is a node–parent relationship that can be extended pairwise to arbitrary subsets of nodes. Formally, let $\Pr(v)$ denote the set of parents of v in a TFPG model G. We can define the *observable consistency* at time t as a relation $\text{OCons}_t \subset \mathcal{P}(V) \times V$ such that $(V', v) \in \text{OCons}$ if and only if $V' = \Pr(v)$, and the observable state of v is consistent with that of all its parents V' based on the map S_t and the failure propagation semantics. The observable state consistency relationship can directly be extended to any set of nodes representing a subgraph of G. In this case, we overload the relationship OCons so that $\text{OCons}_t \subseteq \mathcal{P}(V)$, where for each $V' \subseteq V$, we have

$$V' \in \text{OCons}_t \Leftrightarrow \forall v \in V' \left(\Pr_{V'}(v), v\right) \in \text{OCons}$$

where $\Pr_{V'}(v)$ is the parent of v restricted to V'. The set of maximally consistent set of nodes is denoted Φ_t, where $V' \in \Phi_t$ if and only if

$$V' \in \text{OCons}_t \text{ and } \forall V'' \subseteq V \quad V' \subset V'' \Rightarrow V'' \notin \text{OCons}_t.$$

The set Φ_t can efficiently be computed incrementally based on Φ_{t-1} with a new event e_t. The event e_t corresponds to either a new triggered monitored discrepancy or a timeout event generated when a sensor alarm is not observed at state ON while it is supposed to be based on its current hypothetical state. The underlying procedure will be denoted $\text{UpdateMCO}(\Phi_{t-1}, e_t)$. Note that initially $\Phi_0 = \{V\}$.

Based on the semantics of failure propagation, it is possible to define a constructive notion of *hypothetical consistency* such that, given a consistent hypothetical state $H_t^{V'}$, it is possible to extend this map forward [procedure $\text{BProp}(H_t^{V'}, v)$] by assigning the maximal hypothetical state of the node v based on the hypothetical state of its parents in V' or backward [operation $\text{FProp}(H_t^{V'}, v)$] by assigning the maximal hypothetical state for v' based on the state of its children in V'. The following algorithm outlines the incremental reasoning procedure.

Algorithm 1 The diagnosis procedure $\text{Diag}(\Phi_{t-1}, e_t)$
$\Phi_t \leftarrow \text{UpdateMCO}(\Phi_{t-1}, e_t)$
$HS_t \leftarrow \varnothing$
define
$\text{In}(X) := \{v \in X | (\forall v' \in X)\ (v, v') \notin E\}$
$\text{PSet}(X) := \{v \in V - X | (\exists v' \in \text{In}(X))\ (v, v') \in E\}$
$\text{ODC}(X) := \cup_{v \in X} \text{Reach}(v) - X$
$\text{TSet}(X) := \{v \in V - X | \text{ODC}(X) \times v \cap E = \varnothing\}$
$\text{CSet}(X) := \{v \in \text{TSet}(X) | (\exists v' \in X)\ (v', v) \in E\}$
end define
for all $V' \in \Phi_t$ **do**

$H \leftarrow S_t|_{V'}$
while $\texttt{PSet}(V') \neq \varnothing$ **do**
select v from $\texttt{PSet}(V')$
$H \leftarrow \texttt{BProp}(H, v)$
$V' \leftarrow V' \cup \{v\}$
end while
while $\texttt{CSet}(V') \neq \varnothing$ **do**
select v from $\texttt{CSet}(V')$
$H \leftarrow \texttt{FProp}(H, v)$
$V' \leftarrow V' \cup \{v\}$
end while
for all $v \in V - V'$ **do**
$H(v).\text{state} \leftarrow \texttt{OFF}$
$H(v).\text{terl}, H(v).\text{terl} \leftarrow 0$
end for
$HS_t \leftarrow HS_t \cup \{H\}$
end for
return Φ_t, HS_t

The foregoing diagnosis algorithm returns a set of new hypotheses that can consistently explain the current observed state of the TFPG system. A failure report is then generated from the computed set of hypotheses HS_t. The failure report enlists the set of all consistent state assignments that maximally match the current set of observations. Any observed state that does not match the current hypothesis is considered faulty. A detailed description and analysis of the diagnosis algorithm can be found in [7].

A. Hypotheses Ranking

The quality of the generated hypotheses is measured based on the following three independent factors.

1) *Plausibility* is a measure of the degree to which a given hypothesis group explains the current fault signature. Plausibility is typically used as the first metric for sorting the hypotheses, focusing the search on the failure modes that explain the data that are currently being observed.
2) *Robustness* is a measure of the degree to which a given hypothesis is expected to remain constant. Robustness is typically used as the second metric for sorting the hypotheses, helping to determine when to take action to repair the system.
3) *Failure Rate* is a measure of how often a particular failure mode will occur.

The plausibility metric considers two independent factors, namely, alarm consistency and failure mode parsimony. The alarm consistency factor is defined as the ratio of the active consistent alarms to that of all (currently) identified alarms. The failure mode factor is defined as the ratio of identified failure modes (according to the underlying hypothesis) to the total number of failure modes in the system. This factor is a direct representation of the parsimony principle (fewer failures is more plausible). The hypothesis plausibility metrics are lexicographically ordered (the alarm factor is more dominant).

The diagnoser incrementally selects the current set of hypothesis in an attempt to improve the current plausibility measure. In other words, the diagnoser will update a given hypothetical state map only if such an update can increase the plausibility of the underlying hypothesis. In addition, changes are restricted so that the updated hypothesis remains valid.

IV. Practical Considerations in TFPG-Based Diagnosis

The original TFPG diagnosis system was originally developed about a decade ago for a general class of engineering applications. Since then, major updates have been made to address the complex operational setting of practical applications and the limitations of the underlying execution platforms. In this section, we introduce several modeling and reasoning extensions that have been incorporated into the TFPG diagnosis tool based on experience with real-time vehicle subsystems.

A. Intermittent Faults

Intermittent faults are common in practical engineering systems. They correspond to situations in which some of the failure conditions are observed for a limited time interval. Intermittent faults can be caused by sensor inaccuracy, temporary changes in the system components or its operating conditions, external disturbances, and signal noise. If not considered in diagnosis systems, intermittent faults can lead to a decrease in the diagnosis accuracy or even false diagnosis.

The original TFPG diagnosis algorithm assumes a permanent single state change for any monitored discrepancies. That is, once an alarm is activated, it will remain active throughout the operation cycle of the system. Assuming all faults to be permanent, the steps leading from fault identification to fault isolation is as described earlier in Section III. The situation can be more complicated in the presence of intermittent faults that need to be distinguished and handled accordingly. In some practical settings, intermittent faults are treated as permanent faults until confirmed otherwise. This is possible in situations where the fault recovery procedure is simple and the system is flexible enough to allow various reconfigurations while in operation. In other situations, such an easy recovery is not possible, and the only alternative is to wait until the nature of the fault is confirmed.

In sensor-based diagnosis systems, an intermittent fault will cause a set of alarms to turn OFF after a period of activation. As mentioned earlier, this can lead to a significant decrease in the accuracy of the diagnosis. Such a decrease depends on the average frequency and duration of intermittent faults. It is observed through various experiments that high frequency intermittent faults can lead to information loss, which can then lead to irreversible diagnosis inaccuracy. However, for the more-common situation of low-frequency intermittent faults, it is possible to recover the accurate diagnosis. The recovery can be achieved after all intermittent faults are identified.

The current TFPG diagnosis system implements a backtracking procedure to handle intermittent fault recovery. Using backtracking, diagnosis errors can be corrected by reevaluating the diagnosis output from the time an intermittent is suspected (observed) to the time it is confirmed. The backtracking

procedure is invoked whenever an alarm d changes its state from ON (at time t_i) to OFF (at time t_j). The following data will be needed for the backtracking procedure:

1) the set of all events triggering the diagnoser up to the current time. From this set, we can obtain the set of events triggered between t_i and t_j, denoted $\texttt{DEvents}(t_i, t_j)$;
2) the set Φ_t for all time t at which a new diagnosis event occurred.

The diagnosis procedure $\texttt{Diag}(\Phi_{t-1}, e_t)$ described earlier can be extended to a set of ordered events $E_{i,j} = \{e_{t_i}, \ldots, e_{t_j}\}$, where $j \geq i$ as

$$\texttt{Diag}(\Phi_{t-1}, E_{i,j}) = \texttt{Diag}\left(\texttt{UpdateMCO}(\Phi_{t_i}, E_{i,j-1}), e_j\right)$$

where $\texttt{UpdateMCO}(\Phi_{t_{i-1}}, E_{i,j-1}), e_j)$ is the recursive application of the operation UpdateMCO to the set $\Phi_{t_{i-1}}$ using the set of event $\{e_{t_i}, \ldots, e_{t_{j-1}}\}$ as ordered.

The complexity of the backtracking procedure directly depends on the size of the set $\texttt{DEvents}(t_1, t_2)$ as well as the size of Φ_t, which typically monotonically grows with t_1. In the absence of mode switching and false alarms Φ_t, the size of Φ_t polynomially grows with respect to $|\texttt{DEvents}(0, t)|$. If either false alarms or mode switching is assumed at run time, the growth of Φ_t becomes exponential. Therefore, there could be a significant overhead associated with long-duration intermittent fault or those who appear later in the execution cycle of the system.

To reduce the complexity of the backtracking procedure, potential intermittent alarms (for instance, those with relatively little support from other alarms) are identified, and two different diagnosis paths are considered for both ON and OFF states. This approach can reduce the maximum processing time. However, it will not affect the average processing time, and in general, it will require more memory. Intermittent alarms can be identified using either design criteria, such as alarm failure rate, or through learning from previous failures.

Another approach that can be used to handle intermittent alarms is to delay the diagnosis procedure for a limited time until potential intermittent alarms stabilize. This can be applied in situations where the intermittent duration is relatively short and the system can tolerate the transient state between suspecting an alarm and accurately identifying the state of the alarm. This approach is currently under investigation.

B. Test Alarms

Test alarms are alarms that can be enabled and disabled by the user to help in the diagnosis effort. When a test alarm is disabled, its state is unknown. In contrast, regular alarms are always enabled during the operation cycle of the system, and their state is known as either ON or OFF. Test alarms are used for ground-based maintenance but can also be used at run time to check certain failure conditions (hypotheses) as reported by the diagnoser. In general, a test alarm can be enabled and disabled several times during operation. The value of a test alarm value can only be used in diagnosis if it is enabled. That is, a disabled test alarm is not considered when computing any of the diagnosis metrics.

To handle test alarms in the TFPG diagnosis system, the map DS (defined in Section II) is extended by adding the test alarm type. Accordingly, $\texttt{DS} : D \rightarrow \{\texttt{A}, \texttt{I}, \texttt{T}\}$, where T indicates a test alarm. In addition, the concept of physical state is extended to take into account the enabling condition of the test alarm. When a test alarm is disabled it is treated as an inactive (unmonitored) discrepancy. An inactive discrepancy is assigned a hypothetical state that is (maximally) consistent with the hypothetical states of its parents. On other hand, the hypothetical state of an alarm (active discrepancy) is either equal to its physical state or defined in such a way to be consistent with the hypothetical state of its parents.

An enabled test alarm is more complicated, as it cannot directly be treated as a regular alarm due to the fact that test alarms do not possess the time information associated with regular alarms. In particular, if the test alarm is enabled at time t and responds with an ON state, it is not possible to know the time at which the failure propagated to the associated discrepancy. Accordingly, test alarms are considered time ambiguous. Following the parsimony principle, the time at which a test alarm is triggered is considered in a way that maximizes the plausibility of the underlying hypothesis. In particular, the consistency of test alarms is estimated according to the following conditions.

1) The test alarm d is enabled at time t and responds with an ON state.
   a) If the ON state of d is consistent with respect to the hypothetical state of its parents for a (maximal) time interval $[t_1, t_2]$ such that $t_o \leq t_1$ and $t_2 \leq t$, where t_o is the last activation time of the test alarm, then d is assigned the hypothetical state $(\texttt{ON}, [t_1, t_2])$.
   b) Else, d is assumed an inconsistent alarm and is assigned a hypothetical state that is maximally consistent with its parents as well as the previous activation time of d.
2) The test alarm d is enabled at time t and responds with OFF state.
   a) If the OFF state of d is inconsistent with respect to the hypothetical state of its parents, then the alarm is assigned a maximally consistent hypothetical state with respect to its parents' hypothetical states. Given the discrepancy between the physical and hypothetical state of d, d will then be treated as an inconsistent alarm. Any future activation of d will not affect its inconsistent status.
   b) Else, d is assumed consistent with respect to its parents until future activation of d.

Adding the foregoing conditions to handle test alarms does not have any significant effect on the complexity of the reasoning algorithm. In practice, however, the ambiguity associated with test alarms may allow some false alarms to pass, which can lead to a larger number of hypotheses (explanations) than in a normal situation.

Due to the time ambiguity associated with test alarms, it is desirable in practical applications to distinguish between test alarms and normal ones when considering the plausibility and robustness measures of the underlying hypotheses. In

particular, test alarms are set to have a lower effect (based on a weight defined by the user and certain design factors such as alarm reliability and failure rate) than regular alarms when considered in these two measures.

C. Resource-Limited Reasoning

In many practical applications, the resource allocated to the diagnosis system does not provide enough support for comprehensive reasoning. This is well known to be the case for most model-based diagnosis approaches when applied to typical large-scale engineering systems. This fact has limited the application of model-based techniques in practical domains where system designers typically employ a more affordable and efficient (but somehow limited) associative analysis approach for diagnosis.

The TFPG diagnosis system allows certain aspects of the diagnosis system to be tuned to reduce the run-time reasoning computational requirements. These include the following.

1) Limit the number of hypothesis generated by the diagnosis procedure $\texttt{Diag}(\Phi_{t-1}, e_t)$ by eliminating less-plausible hypotheses throughout the propagation cycle of hypothetical states.
2) Eliminate less-likely hypothetical states based on the following evaluation criteria.
   a) The effect of earlier operating modes are ignored, relative to recent ones.
   b) The monitored discrepancies with low failure rate are considered more relevant than those with high failure rate.

Using simulation, it is possible to have approximate estimations of the effects of changing these values on the accuracy and computation requirements (time and memory) of the diagnosis system such that the diagnoser can be adjusted for specific time and space requirements.

In addition to the foregoing tuning options, the TFPG diagnosis system provides several design-time adjustments to allow trading between various aspects of computational requirements. These include the precompilation of reachability and connectivity information that are frequently used throughout the diagnosis process. This will increase the memory requirement but can significantly enhance the response time of the diagnosis process.

D. Model Verification

TFPG models can incrementally be verified: first in a standalone manner and then in an integrated cosimulation environment prior to the model being automatically translated into an executable code.

1) Standalone Simulation: Standalone test cases can be executed on the engineering desktop via an interactive graphical fault insertion environment (called FPSim) or in a batch mode using simple ASCII files defining each scenario. As a minimum, all logically unique diagnoses that were identified during the up-front testability analysis should individually be exercised against the reasoner model. The TFPG model can be interpreted within the generic modeling environment (GME)[2] to provide a test-case template in which all monitors are initialized to a "pass" state at time 0. The desired monitor triggers can simply be appended to the bottom of the template, and this file can automatically be translated into the columnated format required by the PC-based reasoner. A batch file can then be set up to run all of these test cases from a single point. The resulting log file from each reasoning session can then be compared with the corresponding content of the ambiguity group report. If no testability model or analytical results are available, the user could select each individual failure mode using the interactive FPSim tool, which would then automatically generate the expected fault signature and invoke the TFPG reasoner.

2) End-to-End Cosimulation: The TFPG reasoner can further be exercised in an end-to-end cosimulation environment. A MATLAB or MATRIXx simulation of the fault monitors can be executed in parallel with the TFPG reasoner. Simulated raw sensor streams can be fed into MATLAB or MATRIXx models, with the resulting fault signature automatically sent to the TFPG reasoner for diagnosis. The reasoning results are fed back to MATLAB/MATRIXx for display or for use in decision support logic.

3) Automated Test Case Generation: An automated translation utility was generated to translate a testability model into a TFPG reasoning model and to create a test case for each ambiguity group defined in the testability results by appending the monitors associated with each fault signature to the test-case template at arbitrary time intervals. Additional TFPG utilities then translate these test cases into the reasoner format and execute them against the reasoner. A final executable then creates a log file that concatenates the static ambiguity group data and the dynamic reasoner results in a single log file, simplifying the comparison process. All of these utilities can easily be organized within a batch file to provide a single-step verification process. Note that the nominal test cases remain valid, even if timing or mode constraints were manually added after model translation. However, they would need to be modified to assure the proper order and timing of the monitor sequence and to inject the necessary mode control changes.

An additional number of off-nominal test cases would then typically be added to determine whether the current TFPG model will optimally react to partial data (test voids) or extraneous data (false alarms). These are easily created by adding or deleting monitors from existing nominal test cases.

V. Application to Vehicle Subsystems

The TFPG tool has been used for the modeling, diagnosis, and testability analysis of a generic fighter aircraft fuel transfer system, which is comparable in complexity with models of real-life systems. The system is designed to provide an uninterrupted supply of fuel at a constant rate to the aircraft engines while maintaining the center of gravity of the aircraft. The system is symmetrically divided into left and right parts (top and bottom in the schematic). The four supply tanks [Left Wing (LWT),

[2]The GME is a configurable toolkit developed at Vanderbilt University for creating domain-specific modeling and program synthesis environments [14].

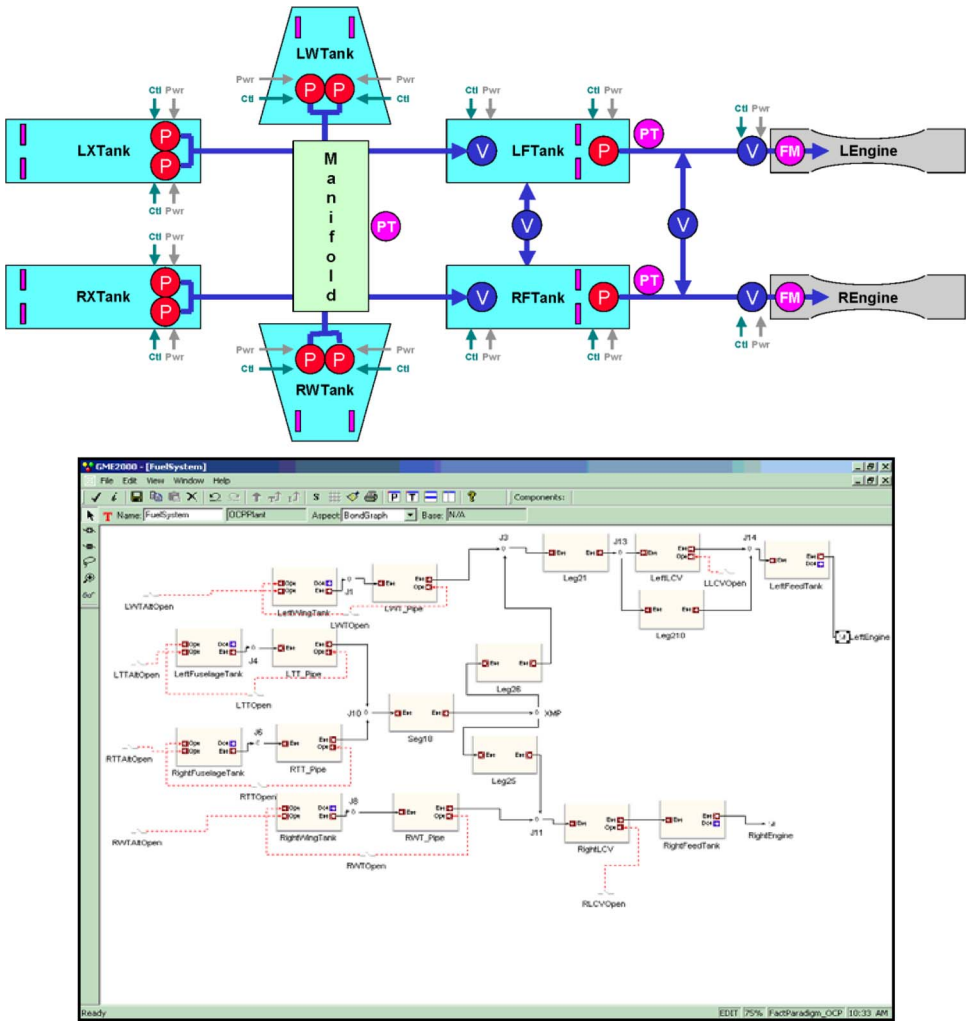

Fig. 2. TFPG model of a generic fuel system.

Right Wing (RWT), Left Transfer (LTT), and Right Transfer (RTT)] are full initially, and so are the two receiving tanks [Left Feed (LFT) and Right Feed (RFT)] that directly feed the engine. During engine operation, fuel is transferred from the supply tanks through a common manifold to the two feed tanks in a sequence determined by the fuel system controller.

The TFPG model is illustrated in Fig. 2. In addition to capturing the failure propagation paths across the main subsystem elements (Left and Right Fuselage Tanks, Left and Right Wing Tanks, Transfer Manifold, and Left and Right Feed Tanks) in great detail, this model also captures the failure propagation paths associated with the power and control elements in the system. This model includes 153 components, 481 failure modes, 1973 discrepancies, 270 alarm monitors, nine modes, and 3409 failure propagation links (555 of them with activation functions described by mode states).

The initial system was exhaustively tested on the data generated from an independent simulator for about 250 fault scenarios. In all cases, the list of "Most Probable Faults" from the reasoner included the "real" fault source(s)/failure mode(s). The worst-case response time on a desktop AMD-Athlon XP(2400+) 2-GHz processor with 448-MB RAM was around 0.1 s, whereas that on a PowerPC 750, 400-MHz, 256-MB RAM, it was around 0.3 s. For more detailed discussion on this case study, see [8].

VI. Conclusion

In this paper, we have presented several practical considerations for diagnosing a general class of engineering systems using TFPGs. We have discussed the problem of intermittent failures and how the TFPG model and reasoning approach have been extended using a backtracking approach. We have also presented the reasoning extensions that were added to handle test alarms, automate model verification, and accommodate limited computational resources.

References

[1] W. Hamscher, L. Console, and J. de Kleer, *Readings in Model-Based Diagnosis*. San Mateo, CA: Morgan Kaufmann, 1992.

[2] S. Abdelwahed, G. Karsai, and G. Biswas, "System diagnosis using hybrid failure propagation graphs," in *Proc. 15th Int. Workshop Principles Diagnosis*, Carcassonne, France, 2004.

[3] A. Misra, J. Sztipanovits, and J. Carnes, "Robust diagnostics system: Structural redundancy approach," in *Proc. SPIE Symp. Intell. Syst.*, 1994, pp. 249–260.

[4] S. Padalkar, J. Sztipanovits, G. Karsai, N. Miyasaka, and C. Okuda, "Real-time fault diagnostics," *IEEE Expert*, vol. 6, no. 3, pp. 75–85, Jun. 1991.

[5] G. Karsai, G. Biswas, and S. Abdelwahed, "Towards fault-adaptive control of complex dynamic systems," in *Software-Enabled Control: Information Technology for Dynamical Systems*, T. Samad and G. Balas, Eds. Piscataway, NJ: IEEE Press, 2003, ch. 17.
[6] V. Brusoni, L. Console, P. Terenziani, and D. T. Dupre, "A spectrum of definitions for temporal model-based diagnosis," *Artif. Intell.*, vol. 102, no. 1, pp. 39–79, Jun. 1998.
[7] S. Abdelwahed, G. Karsai, and G. Biswas, "A consistency-based robust diagnosis approach for temporal causal systems," in *Proc. 16th Int. Workshop Principles Diagnosis*, Pacific Grove, CA, 2005.
[8] S. Ofsthun and S. Abdelwahed, "Practical applications of timed failure propagation graphs for vehicle diagnosis," in *Proc. IEEE Autotestcon*. Baltimore, MD, Sep. 2007, pp. 250–259.
[9] J. de Kleer, "Diagnosing intermittent faults," in *Proc. 18th Int. Workshop Principles Diagnosis*, 2007, pp. 45–51.
[10] I. Koren and Z. Kohavi, "Diagnosis of intermittent faults in combinational networks," *IEEE Trans. Comput.*, vol. C-26, no. 11, pp. 1154–1158, Nov. 1977.
[11] O. Contant, S. Lafortune, and D. Teneketzis, "Failure diagnosis of discrete event systems: The case of intermittent faults," in *Proc. 41st IEEE Conf. Decision Control*, 2002, pp. 4006–4011.
[12] S. Soldani, M. Combacau, A. Subias, and J. Thomas, "Intermittent fault diagnosis: A diagnoser derived from the normal behavior," in *Proc. 18th Int. Workshop Principles Diagnosis*, 2007, pp. 391–398.
[13] A. Correcher, E. Garcia, F. Morant, E. Quiles, and E. Blasco-Gimenez, "Intermittent failure diagnosis in industrial processes," in *Proc. IEEE ISIE*, 2003, pp. 723–728.
[14] J. Sztipanovits and G. Karsai, "Model-integrated computing," *Comput.*, vol. 30, no. 4, pp. 110–112, Apr. 1997.

Sherif Abdelwahed (SM'04) received the Ph.D. degree in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 2002.

From 2000 to 2001, he was a Research Scientist with the System Diagnosis Group, Rockwell Scientific Company, Thousand Oaks, CA. From 2002 to 2007, he was a Research Assistant Professor with the Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN. He conducts research on model-based design and analysis of self-managing computation systems. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Mississippi State University, Mississippi State. He has published over 70 publications. His research interests include modeling and analysis of distributed real-time systems, automated verification, fault-diagnosis techniques, and model-integrated computing.

Dr. Abdelwahed is a member of Sigma Xi.

Gabor Karsai received the B.Sc., M.Sc., and Dr.Techn. degrees from the Technical University of Budapest, Budapest, Hungary, in 1982, 1984, and 1988, respectively, and the Ph.D. degree from Vanderbilt University, Nashville, TN, in 1988.

He is currently a Professor of electrical engineering and computer science and a Senior Research Scientist with the Institute for Software Integrated Systems, Department of Electrical Engineering Computer Science, Vanderbilt University. He has over 25 years of experience in software engineering. He conducts research on the design and implementation of embedded systems, programming tools for visual programming environments, and the theory and practice of model-integrated computing. He has published over 100 papers and is the coauthor of four patents. He has worked on several large projects in the recent past, including one on fault-adaptive control technology that has been transitioned into aerospace applications and another on the model-based integration of embedded systems whose resulting tools are being used in embedded software development tool chains worldwide.

Nagabhushan Mahadevan received the M.S. degree in computer engineering and chemical engineering from the University of South Carolina, Columbia, and the B.E. (Hons.) degree in chemical engineering from Birla Institute of Technology and Science, Pilani, India.

He is currently a Senior Staff Engineer with the Institute for Software Integrated Systems (ISIS), Department of Electrical Engineering Computer Science, Vanderbilt University, Nashville, TN, where his work is focused on model-based diagnosis, model-driven quality-of-service management, and model-based adaptation of software-intensive systems. As the Lead Developer for the "Fault Adaptive Control Technology" tool suite, he has contributed toward product development in support of model-based diagnosis and reconfigurable controllers.

Stanley C. Ofsthun received the B.S. degree in electrical engineering from Rensselaer Polytechnic Institute, Troy, NY, and the M.S. degree in electrical engineering and graduate certificates in artificial intelligence and web content design from Washington University, St. Louis, MO.

He is currently a Technical Fellow with Boeing Integrated Defense Systems, St. Louis. His 25-year career has primarily focused on the practical application of testability, built-in test, integrated diagnostics, and integrated vehicle health management (IVHM) technologies, processes, and tools to a variety of Boeing platforms, including F-15, F/A-18, X-45C, and proprietary space/weapons programs. He has also developed methodologies to integrate these efforts with reliability and safety disciplines and is currently serving as the St. Louis site technical representative on Boeing's Reliability, Maintainability, and Systems Health (RM&SH) steering team.