An Algorithm for Diagnostic Reasoning Using TFPG Models in Embedded Real-Time Applications

Larry Howard, Institute for Software Integrated Systems, Vanderbilt University, (615) 343-7447, howardlp@isis.vanderbilt.edu

ABSTRACT

Embedded diagnostic reasoners require compact modeling representations and efficient reasoning algorithms given limited available computational resources. Timed failure propagation graphs (TFPG) are compact representations used to model failure causes and progressions of conditions that are symptoms of failure occurrence, together with the temporality and likelihood of these symptom progressions and the observation of some of the aberrant conditions. Algorithms for design-time diagnosability analysis using TFPG models have previously been reported, but these algorithms have different design objectives leading to different computational strategies and optimization criteria. This paper presents and discusses an algorithm specifically designed for efficient failure isolation based on reported observations of abnormal conditions that is suitable for use in embedded real-time applications.

Keywords: embedded diagnostics, diagnostic reasoning algorithms

1 INTRODUCTION

While the limitations of purely qualitative modeling representations for failure isolation are well observed, it must be acknowledged that their use continues to be of value when 1) the data required to support more quantitative schemes are unavailable or 2) the computational requirements of quantitative schemes are incompatible with their use in resource-constrained environments. Further, as we make the significant transition from component diagnostics to integrated diagnostics of systems, there may well be a continued use for qualitative representations, such as the timed failure propagation graph addressed in the paper, in representing component interactions under failure conditions. Since we accept that the use of qualitative representations is motivated by practicality, we present an algorithm for diagnostic reasoning using timed failure propagation graphs in embedded applications, less as a contribution to the research literature on model-based diagnostics and more as an example for engineers interested in applying this approach to diagnosis.

Our motivation for the construction of the algorithm we report came from an examination of algorithms for design-time diagnosability analysis. [1][2][3] We observed that these algorithms employed computational strategies that were less concerned with efficiency in failure isolation and more concerned with exhaustive searches of the model space for the computation of various metrics. And while these computational strategies could, and were, used for failure isolation, their efficiency would quickly become an issue as the size of the models increased. We felt that

if we limited the role of an algorithm to simply failure isolation, then a much more efficient algorithm could be realized.

This paper reports our result in designing such an algorithm. It is organized into three primary sections. The first addresses the model representation employed. It presents and discusses the modeling elements, their purposes, and the relationships. The second section presents the algorithm itself. It provides a high-level description, followed by a detailed description of each of the algorithm's constituent phases. The third section provides a brief discussion of the algorithm and of the environment that supports the creation of the modeling representation.

2 MODELING REPRESENTATION

The timed failure propagation graph (TFPG) is a directed graph where the nodes represent *failure modes*, which are fault causes; *discrepancies*, which are off-nominal conditions that are the effects of failure modes; and *monitors*, which are observations of discrepancies. Attributed edges between failure modes and discrepancies in the graph represent causality, and the attributes specify the likelihood and temporality of causation. Attributed edges among discrepancies represent the propagation of effects, and the attributes specify likelihood and temporality of the propagation. Attributed edges between discrepancies and monitors represent observation, and the edges specify likelihood and temporality of observing the discrepancy. In whole, a TFPG represents temporal progressions (paths) of effects that are caused by failure modes, some of which are observed. A failure diagnosis is an explanation of observed discrepancies using these progressions.

The likelihood and temporality attributes on the edges described above have a consistent form. For the specification of causal likelihood, a five level classification scheme is currently employed, with qualitative steps indicating relative degrees of certainty about the occurrence of the respective phenomenon. The specification of temporality is defined by two reference times with respect to the likelihood. From 0 to the first reference time, t_{min}, the phenomenon does not occur. From t_{min} to the second reference time, t_{max}, the phenomenon occurs with the specified likelihood. After t_{max} the phenomenon occurs with certainty.

A given discrepancy can be caused by multiple failure modes, or else it can lie at the intersection of multiple propagation paths. Multiple edges entering a discrepancy are interpreted using "OR" semantics. By this we mean that any one of the entering edges could supply the cause of the discrepancy. Less frequently "AND" semantics are required, meaning that more than one of the edges entering a discrepancy must supply failure causes for the discrepancy to occur. To support these semantics, there is an "And" conjunctive node. Edges entering this node must each supply a failure cause before the outbound edge(s) of the node will propagate an effect.

Nodes and edges in a TFPG representation can be present or absent under particular conditions, such as flight phases or operating modes. The representation supports the definition of these conditions through special purpose nodes, and the supporting modeling environment supports associating failure propagation nodes and edges with these conditions using inclusion and exclusion classifiers. This process is called *conditionalization*.

When using a TFPG representation to model failure propagation in complex systems, it is desirable to use hierarchy to manage system complexity. Hierarchy is supported through typed containers, each of which is a TFPG model. The use of typed containers permits the definition of modeling concepts (modes, for example) that are appropriate only within a certain level, or levels, of the hierarchy. These containers provide ports, by which propagation paths can be brought in to and out of the container (model) from higher levels in the hierarchy. A model can contain lower level containers, and propagation paths within the containing model can be attached to their ports. The traversal of paths represented in a TFPG can thus move up and down the containment hierarchy. It is expected that the containment hierarchy in a TFPG model will mirror the physical organization of the system being modeled.

References are a notational convenience much like "pointers" in a conventional programming language. Through their use, a model element can be defined in one model and then used to form associations in another model, potentially at a different level of the hierarchy. References and hierarchy, while important to the modeling task, are not essential to the TFPG representation nor are they important to the reasoning algorithm. Therefore, nothing further will be said about them here.

Figure 1 gives an example of a TFPG model. This example shows the two types of ports used to carry propagation effects into and out of models: *flows* and *signals.* The former are used to represent the propagation of effects by means such as energy exchanges, while the latter are used to represent all effects carried along signal pathways. This distinction is useful in constructing understandable models of failure propagation with respect to interactions among components under failure conditions, but it is not essential to the task of failure isolation, and so it too will not be discussed further.

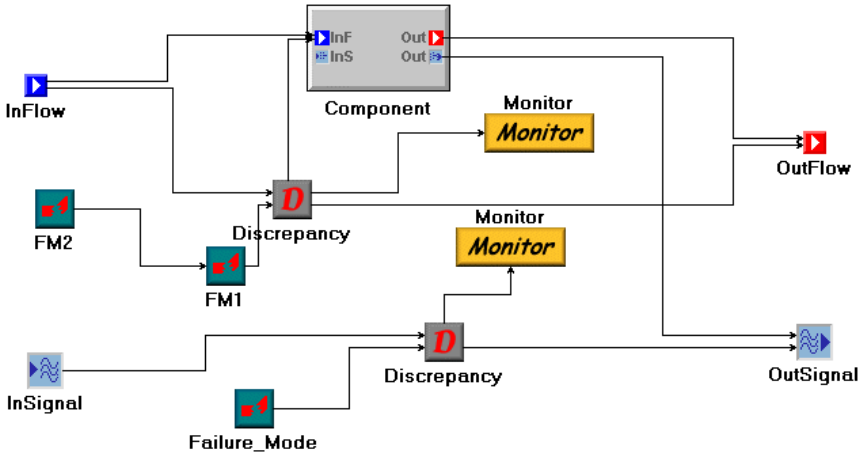

Figure 1: A Timed Failure Propagation Graph (TFPG) Model

3 ALGORITHM

At the highest level, the diagnostic algorithm consists of two phases:

1. Graph traversal, segmentation, and reconstruction.
2. Hypothesis generation and scoring.

The first of these phases is graph-theoretic and deductive. It begins by traversing the TFPG from reported observations in a depth-first search for either failure modes or any node in the graph that has been reached by an earlier traversal. During traversal, segmentation of the graph is performed and reached nodes are marked. Segmentation is a performance optimization used to eliminate re-traversal of the graph by preserving descriptions of significant path segments. Examples of such segments are monitors to monitors, monitors to failure modes, and monitors to "AND" conjunctives.

Once any new segments have been identified, reconstruction is performed. The goal of reconstruction is to eliminate as many failure causes for observations as possible using temporality, missing observations, and other evidence. This is the deductive aspect of the algorithm's first phase. Reconstruction results in ordered sets of path segments where the starting node of the first segment of each set is an observation and the final node of the last segment is a failure mode. In effect, this represents the set of all possible explanations for every observation.

The second phase of the algorithm is set-theoretic and heuristic. This phase begins by determining a mapping between failure modes and the observations they explain, given the results of reconstruction. Using this mapping, hypotheses are generated using the Law of Parsinomy; i.e., when creating explanations, simplier ones are preferable to more complicated ones, and those agreeing with existing explanations are preferable to those that require existing explanations to be abandoned. The procedure for applying this heuristic is a follows:

1. Form of a set of all observations to be explained.
2. Determine the reconstruction that explains the largest subset of the observations to be explained.
3. Generate a failure hypothesis for this subset of observations with the failure cause(s) of the reconstruction as the explanation.
4. Add to this failure hypothesis any other reconstructions that explain the same subset of observations as alternative explanations.
5. Eliminate this subset of observations from those to be explained.
6. While any observation remains unexplained and there is a possible explanation, repeat steps 2-5.

Once the failure hypotheses are generated, each explanation in each failure hypothesis is scored. The failure likelihood of an explanation is the product of the failure likelihoods of each failure cause (failure mode). The causal likelihood is the minimum causal likelihood along the propagation paths from failure modes to observations. These scores are provided as independent terms for evaluating the explanations constituting a failure hypothesis.

3.1 Inputs and Outputs

The input to the reasoning algorithm is a set of monitor reports. Each monitor report provides the following information, with the identifier in parenthesis being the name of the TFPG attribute of the corresponding monitor:

- Monitor GUID (*GUID*)
- Time of detection (*DRactiveTime*)
- Active/Inactive (*DRactive*)

There are a few different styles of reporting that are supported by the current algorithm. One is for monitor reports to contain the status of every monitor. Another is for reports to contain the current set of active monitors. Still another is for reports to contain the set of monitors whose status has changed since the previous report. Regardless of which style is used, each time a diagnosis is performed there is a set of currently active monitors. We will refer to this set as $O_{active} \subseteq M$, where M is the set of defined monitors.

The output of the reasoning algorithm is a diagnosis consisting of a set of failure hypotheses DR for a partition G of O_{active}. Let $O \hat{I} G$, then a failure hypothesis $fh=(O, E_O)$, where E_O is a set of independent (that is, alternative) explanations of O. $\forall e \hat{I} E_O$, $e=(fl, cl, O_{secondary}, FC)$, where fl is an aggregate failure likelihood, cl is an aggregate causal likelihood, $O_{secondary}$ is a set of secondary observations [1], and FC is a set of failure causes for the explanation.

3.2 Detailed Descriptions

The following subsections give detailed descriptions of each of the steps in the algorithm: traversal, segmentation, reconstruction, hypothesis generation, and scoring. Since traversal and segmentation occur coincidently, they are presented together.

As a starting point, the following definitions and properties describe the TFPG representation as a whole:

1. $TFPG=(N_{TFPG}, E_{TFPG})$, where N_{TFPG} is the set of nodes and E_{TFPG} is the set of edges constituting the directed graph.
2. $N_{TFPG}=(FM,M,D,A)$ where FM is the set consisting of nodes whose type is Failure Mode, M the set of Monitors, D the set of Discrepancies, and A the set of "And" conjunctives.
3. Let $D_{monitored} \subseteq D$ define the Discrepancies which are monitored; that is, $\forall d \hat{I} D_{monitored}$, $\exists m \hat{I} M$ and $\exists e \hat{I} E_{TFPG}$ such that $e=(d, m)$.

3.2.1 Traversal and Segmentation

The result of these steps is a set S of path segments (directed subgraphs) where each $s \hat{I} S$ has the following properties:

1. $s=(N, E)$.
2. $N=\{n_1, n_2,\ldots n_n : n_i \in N_{TFPG}\}$. n_1 is the originating node and will be written as $n_1(s)$. $n_1 \in FM \cup M \cup A$. n_n is the terminating node and will similarly be written $n_n(s)$. If s intersects another segment, then $n_n \in N_{TFPG}$; otherwise, $n_n \in FM \cup M \cup A$.
3. $E=\{e_1, e_2,\ldots e_n : e_i \in E_{TFPG}$ and $e_i=(n_i,n_{i+1})\}$; that is, the ith element of E is the edge connecting the ith node in N with its successor on the path.

[1] Secondary observations are observations made upstream of an "And" conjunctive that belong to an explanation for observations made downstream of the conjunctive. Since only some of the failure causes of the explanation account for these upstream observations, they are considered secondary to the explanation.

Let $O_{reported}$ be a set of monitor reports received at time τ. $\forall o \in O_{reported}$

1. Identify the $m \in M$ where *GUID(m)* is the same as the monitor GUID given in *o*.
2. Set *DRactive(m)* and *DRactiveTime(m)* to the reported state of *o*.

Let $O_{active} \subseteq M$ be the set of monitors whose reported state is active at time τ. $\forall o \in O_{active}$ traverse TFPG as follows:

1. If *o* has been previously traversed, then stop this traversal.
2. Identify the set of edges $E \subseteq E_{TFPG}$ terminating at *o*.
3. $\forall e \in E$, *Traverse*(*e*).
4. Mark *o* as having been traversed.

Traverse is a recursive function that is a depth-first search consisting of the following steps:

1. Let *n* be the origin of *e*.
2. If $n \in FM \cup A \cup D_{monitored}$, then
    a. Create a new segment from the start of this traversal to *n* and add to *S*.
    b. If *n* has not previously been traversed, start a new traversal originating at *n*, else stop this traversal.
3. Else if *n* has been previously traversed
    a. If *n* intersects with an existing segment, then form a segment from the start of this traversal to *n* and add to *S*.
    b. Stop this traversal.
4. Identify the set of edges $E \subseteq E_{TFPG}$ terminating at *n*.
5. Mark *n* as having been traversed.
6. $\forall e \in E$, *Traverse*(*e*).

The set *S* of path segments, the markings indicating prior traversal, and the status of reported monitors constitute the persistent state of the Diagnostic Reasoner.

3.2.2 Reconstruction

The result of this step is a set *R* of "reconstructions" where each $r \hat{I} R$ has the following properties:

1. $r=(S_r, LastObservedTime, PropagatedTime)$.
2. $S_r = \{s_1, s_2, \ldots, s_n: s_i \hat{I} S\}$, where s_1 is the originating segment and will be written as $s_1(r)$. s_n is the terminating segment and will be written as $s_n(r)$. $n_1(s_1) \in D_{monitored}$ and $n_n(s_n) \in FM \cup A$.
3. *LastObservedTime* is defined as the *DRactiveTime(m)* of the last segment where $n_1(s_1) \in D_{monitored}$ and *m* is the monitor of $n_1(s_1)$.
4. *PropagatedTime* is the sum of the t_{min} attribute on all edges from s_i to s_n where s_i corresponds to *LastObservedTime*.

Given *S* and O_{active}

1. $\forall o \in O_{active}$

a. Identify $S_o \subseteq S$ where $\forall s \in S_o$, $n_1(s)=o$.
b. $\forall s \in S_o$, *Reconstruct(s, $R_{current}$)*, where $R_{current}$ is a reconstruction to which *s* will potentially be added.
c. Preserve the set of successful reconstructions as *R*.

Reconstruct is a recursive function that assembles path segments as follows:

1. Check that each node and edge in *s* are present for the current mode of conditionals to which they belong. If not, then discard $R_{current}$ and return.
2. $PropagatedTime(R_{current}) = PropagatedTime(R_{current}) + \sum(t_{min}(e), \forall e \in E(s))$.
3. If $n_n(s) \in D_{monitored}$, then for the corresponding monitor *m*
   a. If $m \in O_{active}$, then
      i. If $PropagatedTime(R_{current}) > LastObservedTime(R_{current}) - DRactiveTime(m)$ then, discard $R_{current}$ and return.
   b. Else if $CausalLikelihood(n_n(s), m)=Certain$, then discard $R_{current}$ and return.
4. If $n_n(s) \in A$, then
   a. Identify $S_A \subseteq S$ where $\forall a \in S_A$, $n_1(a)= n_n(s)$.
   b. $\forall a \in S_A$, *Reconstruct(a, $AR_{current}$)*.
   c. Preserve the set of successful reconstructions as AR_a.
   d. If $AR_a=\emptyset$, then discard $R_{current}$ and return.
   e. Else, terminate $R_{current}$ at *s* and return.
5. Identify $S_T \subseteq S$ where $\forall t \in S_t$, $n_1(t)= n_n(s)$.
6. If $S_T=\emptyset$, then
   a. If $n_n(s) \in FM$, then terminate $R_{current}$ at *s* and return.
   b. Else, discard $R_{current}$ and return.
7. Else, $\forall t \in S_T$, *Reconstruct(t, $R_{current}$)*.

3.2.3 Hypothesis Generation

The output of this step is a set *DR* of failure hypotheses that are the unscored result of the diagnostic algorithm.

Given *R* and O_{active}

1. Create a mapping *C*, where $\forall c \in C$, $c=(t, O_t)$ such that $\exists r \in R$ where $t=n_n(s_n(r))$ and O_t is the set of observations explained by *t*.[2]
2. Find the $c \in C$ where $O_t(c) \cap O_{active}$ has the largest number of members.
3. Create a failure hypothesis *fh* and set $O(fh)=O_t(c)$.
4. Create an $e \in E_O(fh)$ and
   a. If $t \in FM$, then $FC(e)=\{t(c)\}$.
   b. Else
      i. Identify $S_t \subseteq S$ where $\forall s \in S_t$, $n_1(s)= n_n(t)$.
      ii. $\forall s \in S_t$
         1. If $n_n(s) \in D_{monitored}$, then add $n_n(s)$ to $O_{secondary}$.

[2] $t \in FM \cup A$.

2. Else if $n_n(s) \in FM$, then add $n_n(s)$ to $FC(e)$.
3. Else set t to $n_n(s)$ and go to i.

5. Identify $A \subseteq C$, where $\forall a \in A$, $a \neq c \wedge O_t(a) = O_t(c)$ and then add each $a \in A$ to $E_O(fh)$ using the procedure identified in step 4.
6. $\forall o \in O_t(c)$ remove o from O_{active}.
7. Remove c from C and $\forall a \in A$, remove a from C.
8. Add fh to DR.
9. While $O_{active} \neq \varnothing$ and then $C \neq \varnothing$, repeat steps 2-8.

3.2.4 Scoring

Given DR and R,

1. For each $fh \in DR$ and each $e \in E_O(fh)$
   a. $fl(e) = \prod(FailureLikelihood(fc), \forall fc \in FC(e))$.[3]
   b. $\forall fc \in FC(e)$ and $\forall o \in O(fh)$
      i. Identify $r_{fco} \in R$ such that $o = n_1(s_1(r_{fco}))$ and $fc = n_n(s_n(r_{fco}))$.
      ii. $cl(e) = Certain$
      iii. $\forall s \in r_{fco}$ and $\forall d \in E(s)$
         1. If $CausalLikelihood(d) < cl(e)$, then $cl(e) = CausalLikelihood(d)$.[4]

The failure hypotheses having been scored, the algorithm is concluded.

4 DISCUSSION

The algorithm reflects a number of optimizations that reduce computational requirements. Examples are the introduction of segmentation into the traversal phase and the integration of deductive reasoning into the reconstruction phase. The first eliminates the re-traversal of any node of the graph that has previously been traversed. This includes paths that intersect an existing path segment. The second eliminates as many possible explanations as possible prior to the hypothesis generation phase, thereby reducing its computational requirements.

The use of the Law of Parsimony in the hypothesis generation phase of the algorithm reflects the use of the diagnostic result as input to the maintenance task. While a complete set of hypotheses could be generated as an output of the phase, making the diagnostic result more robust, its was considered important to focus the diagnostic result for presentation to maintenance personnel to as small a set as possible. As a consequence, it is possible that the algorithm can fail to report a possible, but highly unlikely, failure cause in its diagnosis. Future applications will tell whether it is better to provide a more robust output of the embedded diagnostics that is post-processed at the time of use by maintainers.

---

[3] When $\exists e \hat{I} E_O$ containing multiple, independent failure causes, the likelihood of their mutual occurrence is the joint probability of their individual occurrence. The *FailureLikelihood* attribute of failure modes is a probabilistic term; i.e., the incidence of failure during some unit time interval over the total incidence of failures for that time interval.

[4] For the symbolic definition of *CausalLikelihood*, the relation "<" is defined on the corresponding ordinal values of the symbols, for which *Certain* has the highest value.

As a final point of discussion, let us turn to the issue of the construction of the TFPG representation, upon which the effectiveness of the diagnostics ultimately rests. Given the qualitative nature of fault models such as the TFPG, and the fact that human experts are asked to construct the model representation based on their understanding of complex system phenomena, it is important to provide as much assistance to the modeling task as possible.

The modeling environment used to support TFPG modeling is an extensible framework that allows general purpose and domain-specific extensions.[4][5] Domain-specific modeling paradigms, such as the one for TFPG, are described to the framework by means of meta-models. To support TFPG modeling, we have extended this framework with general-purpose tools for reachability and path analyses. Using these tools, the modeler can determine basic implications of modeling decisions. More specific feedback, however, requires diagnostic reasoning itself. In providing this feedback, we faced two alternatives. The first was to create a design-time reasoner implementation specifically built to interface with the modeling environment. This would have been an easier implementation path, but it presented a serious pitfall. What if the diagnostic results from this reasoner differed from those of the run-time reasoner? The alternative was a single reasoner implementation that could be used in either the emdedded or modeling environment contexts. (See Figure 2.) Our implementation of the reasoner employs an application programming interface (API) for access to the models. When in the emdedded context, this API is satisfied with a compiled run-time representation of the models. When interoperating with the modeling environment, the API is satisfied with calls to a component integration interface of the modeling environment. In this way, the modeler can be confident that feedback on modeling decisions while in the modeling environment will be consistent with the results of the embedded reasoner.

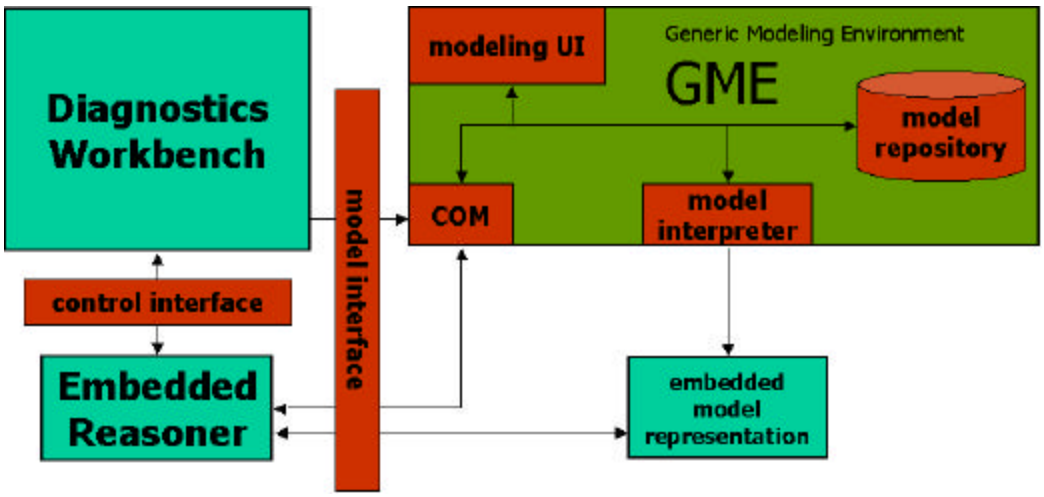

Figure 2: Embedded Reasoner Integration with Modeling Environment

The Diagnostics Workbench component shown in Figure 2 is another valuable extension to the modeling environment that supports the analysis of recorded, simulated, and live observation steams. Using this capability, the modeler can organize complex failure scenarios and use them to validate or refine the models.

5 SUMMARY

We have presented an algorithm for failure isolation based on timed failure propagation graphs that is suitable for use in embedded applications. We have also presented and discussed issues of supporting the creation and validation of the underlying modeling representation. As the usefulness of qualitative models for diagnosis, such as the TFPG, is likely to continue for as long as quantitative schemes have computational requirements exceeding the resources available in many embedded application environments, and until such time as quantitative schemes have demonstrated their efficacy for integrated diagnosis, we consider the availability of efficient diagnostic algorithms based on qualitative models to be of benefit.

6 ACKNOWLEDGEMENTS

We gratefully acknowledge the support of this work by The Boeing Company and the Defense Advanced Research Projects Agency (DARPA).

7 References

[1] Misra A., Sztipanovits J., Underbrink A., Carnes J.: **Diagnosability Analysis and Robust Diagnostics with Multiple Aspect Modeling**, Abstracts of the NASA Workshop on Model-Based Diagnosis and Monitoring; Pasadena, CA, January, 1992.

[2] Misra A., Sztipanovits J., Underbrink A., Carnes J., Purves B.: **Diagnosability of Dynamical Systems**, Third International Workshop on Principles of Diagnosis, Rosario, WA, October, 1992.

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

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

[5] Ledeczi A., Maroti M., Bakay A., Karsai G., Garrett J., Thomason IV C., Nordstrom G., Sprinkle J., Volgyesi P.: **The Generic Modeling Environment**, Workshop on Intelligent Signal Processing, accepted, Budapest, Hungary, May 17, 2001.