Graph-Based Ontology-Guided Data Mining for D-Matrix Model Maturation

Shane Strasser, John Sheppard, Michael Schuh, Rafal Angryk, Clemente Izurieta
Department of Computer Science, Montana State University
357 EPS Building
Montana State University
Bozeman, MT 59717
406-994-4780
shane.strasser@cs.montana.edu

Abstract—In model-based diagnostic algorithms, it is assumed that the model is correct. If the model is incorrect, the diagnostic algorithm may diagnose the wrong fault, which can be costly and time consuming. Using past maintenance events, one should be able to make corrections to the model in order for diagnostic algorithm to correctly diagnosis faults. In this paper, a maturation approach is proposed which uses the graph-theoretic representations of Timed Failure Propagation Graph (TFPG) models and diagnostic sessions based on recently standardized diagnostic ontologies to determine statistical discrepancies between that which is expected by the models and that which has been encountered in practice. These discrepancies are then analyzed to generate recommendations for maturing the diagnostic models. Maturation recommendations include identifying new dependencies and erroneous or tenuous dependencies. [1] [2]

TABLE OF CONTENTS

1. INTRODUCTION

Timed failure propagation graphs (TFPG) were first introduced in 1994 to provide improved robustness in fault diagnosis by analyzing temporal relationships in alarm events [1], [2]. Several diagnostic algorithms have been developed to utilize these TFPG models by determining the most likely fault occurrence given a set of alarms that have been triggered [3], [4]. TFPG models have also been extended to include model dependency constraints on the propagation links by Abdelwahed in 2004 [5]. These extended models, referred to as a Hybrid Failure Propagation Graphs (HFPG), allow the model to operate in in various operational modes. The different operational modes allow alarms to be either enabled or disabled. In [6], the authors proposed a hierarchical diagnosis approach for complex causal systems. In their approach, the system is partioned into a set of local subsystems, each of which represent a sub-graph of the entire system. All of the local subsystems are then contained within a global system that obtains a globally consistent diagnosis of the entire TFPG system. Figure 1 gives an example TFPG model. In the example, the nodes labeled with F1, F2, F3, and F4 represent faults in the TFPG model. The labels D1 through D11 denote nodes that represent discrepancies. Monitored discrepancies, or alarms, are represented by nodes labeled by M2, M3, M9, M10, and M11. Monitors allow the reasoner to detect if a discrepancy has been triggered or turned ON. By looking at which alarms are ON and OFF, the reasoner will diagnosis the most likely fault that was triggered.

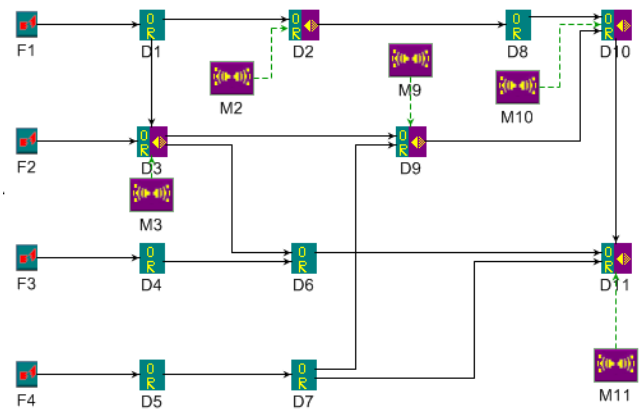


Figure 1. A sample TFPG mode. Nodes that are labeled F1, F2, F3, and F4 are faults. Nodes labeled D1 through D11 are discrepancies, and nodes labeled M2, M3, M9, M10 and M11 are monitors of discrepancies. Discrepancies with monitors are often referred to as alarms.

One of the difficulties with using TFPG models for fault diagnostics is that diagnosis performance is dependent on how accurate the TFPG model is. A bad TFPG model will result in poor diagnosis from the reasoner. The problem is increased when a hierarchical diagnosis approach is used because it is difficult to know what relations should exist

[1] 978-1-4244-7351-9/11/$26.00 ©2011 IEEE.
[2] IEEEAC paper #1383, Version 12, Updated Janurary 11, 2011

between the different subsystems or how the dynamical system will behave in different environmental conditions [7]. If there is an error in the model, then the reasoner's likelihood of diagnosing the correct fault in the system will decrease. This will cause an increase in time, money, parts, and labor in the maintenance of the modeled system since the corrective action will not be known [7]. For example, the TFPG model in Figure 2 has had the edge from D7 to D9 deleted. In this faulty model, the reasoner will not be able to accurately diagnosis fault F4 should have occurred. The reasoner will find that alarms M9, M10, and M11 were all triggered. However, fault F4 will not be diagnosed as the fault since the model does not have a relationship between fault F4 and alarms M9, M10, and M11, increasing the maintenance time since the true fault has to be located by alternative means [9].

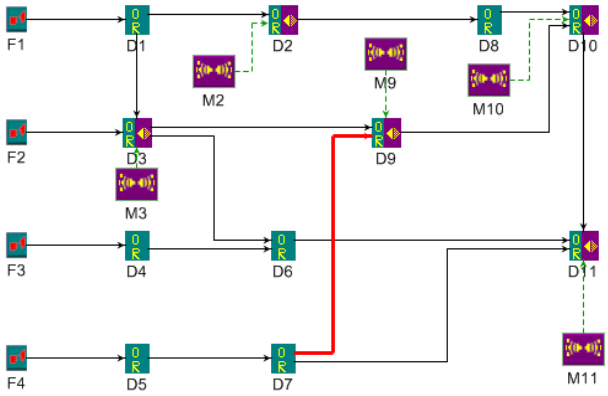


Figure 2. A faulty TFPG model. The link from discrepancy D7 to discrepancy D9 has been removed.

To determine whether the reasoner diagnosed the correct fault, one must compare the reasoner's diagnosis with the actual fault found by alternative means. By storing past reasoner history and maintenance history, the later of which contains the correct fault diagnosis, one can compare the two history sessions and look for any discrepancies between the reasoner's history and the maintenance history. If there is a discrepancy between the two histories, then we know that the reasoner misdiagnosed a fault. If the reasoner is misdiagnosing a particular fault a large number of times, then there could an error in the TFPG model. The discrepancies between the reasoner and maintenance history can then be used to modify the TFPG model such that the reasoner will output the fault that has been occurring [7]. Therefore, we propose a maturation process that is able to look at prior maintenance events and use that information to make changes to the TFPG model in order to improve the accuracy of the model. In the example in Figure 2, we observed that fault F4 is not diagnosed as the fault when alarms M9, M10, and M11 are triggered. Using this information, a change should be made to the model so that given those alarm sequences, fault F4 is diagnosed as the true fault. In this scenario, a link between either the fault F4 node or discrepancy nodes D5 or D7 should be connected to alarm M9.

TFPG maturation is a difficult problem [8]. First, all of the reasoner diagnosis history and maintenance history is needed to be able to locate where the reasoner is misdiagnosing a fault. These data sources are often stored in heterogeneous systems and therefore makes retrieval and analysis of the data difficult. For example, maintenance data is usually stored, but many times the most important aspect of the data is human entered text fields, which are difficult to interpret automatically. Reasoner and maintenance data could also be stored among many repositories, which makes pulling them all together difficult [9].

One possible way to overcome these difficulties is to use a domain ontology to join the different data sources together in a meaningful way. In prior work, Wilmering and Sheppard suggested an approach to utilizing domain ontologies as a means to focus and filter data analysis in knowledge discovery [10]. The specific focus of that work was utilizing the ontologies to guide the process by which diagnostic models could be matured over time. That paper proposed using a method such as the *Apriori* Algorithm to discover new relationships within historical maintenance data that could be used to determine diagnostic relationships, improved probability estimates, or better specification of test processes.

In this paper, we describe an extension of this work in which diagnostic models and historical diagnostic session data are mapped to two ontologies derived from IEEE Std 1232 Standard for Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE) and Std 1632.2 Software Interface for Maintenance Information Collection and Analysis (SIMICA): Maintenance Action Information (MAI) [11], [12]. Specifically, the AI-ESTATE D-Matrix Inference Model provides a semantic definition of information used to define diagnostic models based on diagnostic dependencies while the AI-ESTATE Dynamic Context Model provides a semantic definition of the information typically used by diagnostic reasoners during online reasoning to track test results, record inferences, and recommend hypotheses. The SIMICA MAI model defines information elements associated with maintenance history. These two models, defined by the IEEE using the EXPRESS language, have been redefined using the web ontology language (OWL) [13], [14]. D-matrix models and diagnostic sessions are then mapped to the ontologies and represented using OWL-based instance formats.

The maturation approach uses the graph-theoretic representations of the models and sessions to determine statistical discrepancies between that which is expected by the models and that which has been encountered in practice. These discrepancies between actual maintenance events and what the TFPG reasoner reported are then stored and used by our TFPG maturation approach. From this, we are able to recommend changes, such as adding or removing links between discrepancies. We are also able to track and estimate false alarms and non-detect rates. Once the changes

have been made the reasoner should be able to correctly diagnosis the fault.

In developing our TFPG maturation approach, we looked at two different scenarios. First, we looked at how to mature which alarms are monitoring which faults. This scenario is very similar to the example previously described. The second maturation scenario we studied was how different alarms causally depend on each other. For example, one could be given an alarm sequence in the order of A1, A2 and A3. In a different case, we observe the same alarms, but in a different order, such as A1, A3 and A2. If the reasoner is diagnosing the wrong fault in these cases, then there is some erroneous relationship in our model. In our work, we focused mostly on the first scenario. However, we have developed and are currently testing an algorithm for the second scenario.

2. TIMED FAILURE PROPAGATION GRAPHS

A timed failure propagation graph (TFPG) model is a directed graph in which each vertex represents a failure node or discrepancy [1], [2]. Failure nodes represent faults in the target system and discrepancies are causal nodes that are affected by failure nodes. Discrepancies can be monitored or unmonitored. Monitored discrepancies are often referred to as alarms. The edges between the nodes represent the effect of failure propagation over time in the underlying system that is being modeled. Formally, this is represented as TFPG = *(F, D, E, M, ET, EM, DC, DS)* where:

- F is a set of failure nodes
- D is a set of discrepancy nodes
- $E = V \times V$ is a set of edges, where $V = F \cup D$
- M is a nonempty set of system modes
- ET: $ET(E) \rightarrow Int$, Mapping for each edge in E where Int denotes finite time intervals on each edge.
- EM: Map that associates every edge in E with a set of modes in M
- DC: $DC(D) \rightarrow$ {AND, OR}, Map which defines the type of each discrepancy as either an AND or an OR discrepancy
- DS: $DS(D) \rightarrow$ {A, I}, Map defining the monitoring status of the discrepancy as either active (A) for discrepancies attached to monitored alarms or inactive (I) otherwise

The set of discrepancies that are monitored are defined by the map DS. The map ET associates with each edge e in E a minimum and maximum time for the failure to propagate along the edge. EM associates each edge with a subset of the system modes at which the failure can propagate along the edge. DC defines if each discrepancy is an AND or an OR node. The goal of a diagnostic algorithm is to find a hypothetical state that tries to explain the physical system based on the observed system [3]. In our TFPG maturation approach, we only deal with TFPG models that contain only OR discrepancies and operate in only one mode.

The D-matrix is a matrix representation that relates the faults and the discrepancies that monitor or observe those faults. We can also formally represent it as the following. Let F represent a set of faults. Let D represent the set of discrepancies. Assume each $f_i \in F$ is a Boolean variable such that $eval(f_i) \in \{0,1\}$ and each $d_j \in D$ is also a Boolean variable such that $eval(a_j) \in \{0,1\}$. Then a *diagnostic signature* is defined to be the vector

$$f_i = [eval(d_1),...,eval(d_{|T|})] \quad (1)$$

Where

$$eval(d_j) = \begin{cases} 1 & \text{if } d_j \text{ detects } f_i \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

and $f_i[j]$ is the j^{th} element in vector f_i. A *D-matrix* is then defined to be the set of diagnostic signatures d_i for all $d_i \in D$ [15]. Rows represent faults and columns represent discrepancies. The ith column corresponds to discrepancy i in the TFPG model. The matrix corresponding to D-matrix for Figure 1 would then be as follows:

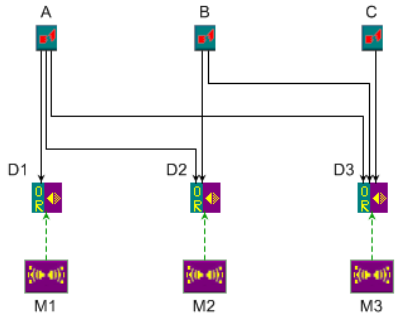
	$D1$	$D2$	$D3$	$D4$	$D5$	$D6$	$D7$	$D8$	$D9$	$D10$	$D11$
$F1$	1	1	1	0	0	1	0	1	1	1	1
$F2$	0	0	1	0	0	1	0	0	1	1	1
$F3$	0	0	0	1	0	1	0	0	0	0	1
$F4$	0	0	0	0	1	0	1	0	1	1	1

However, in most situations, the only discrepancies that are included or shown in the D-matrix are the monitored ones. The D-matrix for Figure 1 would be represented as the following where the columns are the alarms in numerical order would be defined as follows:

	$D2$	$D3$	$D9$	$D10$	$D11$
$F1$	1	1	1	1	1
$F2$	0	1	1	1	1
$F3$	0	0	0	0	1
$F4$	0	0	1	1	1

D-matrices do not fully represent TFPG models because they do not capture the temporal relationships. Nevertheless, the representation of the model is easy to manipulate. Given a D-matrix which relates the faults and alarms, we can actually find the logical relationship between the alarms by computing the logical closure of the matrix [16]. This is done by determining which attributes have a parent set that is a subset of another attribute's parent set. Let a_i be an alarm that monitors faults f_i and let a_j be an alarm that monitors faults f_j. We can represent this as $f_i \rightarrow a_i$ and $f_j \rightarrow a_j$. If f_i is a subset of f_j, then f_j contains f_i and $f_j \rightarrow a_i$. If a_j is

true, f_j must also be true. This means a_i must be true and therefore $a_i \rightarrow a_j$ [16], [17]. Take for example the TFPG model and the corresponding D-matrix which shows the relationship of faults and alarms in Figure 3. The faults that are observed by M1 are a subset of the faults that are observed by alarm M2. Similarly, the faults that are observed by M2 are a subset of the faults that are observed by alarm M3. The TFPG model that would result can be seen in Figure 4.



	M1	*M2*	*M3*
A	1	1	1
B	0	1	1
C	0	0	1

Figure 3. A simple TFPG model and the corresponding D-Matrix. A, B, and C are faults while M1, M2, and M3 are alarms.

Using the original D-matrix one can easily find the logical closure matrix of the discrepancies. This matrix relates discrepancies to other discrepancies. Again, similar to the first D-matrix, a 1 in the *i*th row and *j*th column means that the *j*th discrepancy observers the *i*th discrepancy. The D-matrix in Figure 4 is the result of taking the closure of the original D-matrix.

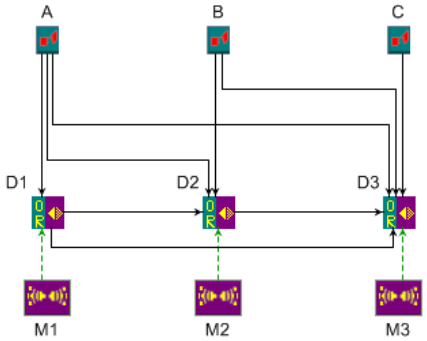
In addition, [16] showed that using the logical closure matrix, the transitive links between discrepancies can be removed by using logical relationships. This process is called taking the logical unclosure.

After finding the logical unclosure of the matrix, the logical NOT is taken over the subset and performing an AND between the subset of parents for an alarm and the set of alarms. In doing so, the transitive edges in the TFPG model and the corresponding D-matrices are removed. The TFPG model in Figure 5 and the corresponding D-matrix is the model that results after taking the logical unclosure of the TFPG model in Figure 4.

This unclosed D-matrix is then able to show the first order dependencies between the discrepancies. In our experiments, we will assume that we only have access to the D-matrix models as they are a universal data representation of TFPG models.

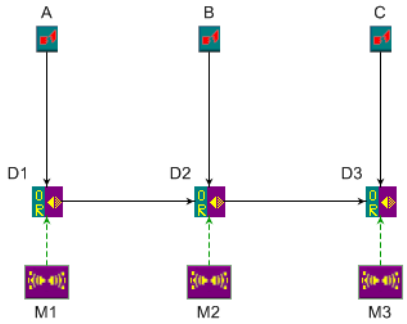
3. Related Work

The idea behind diagnostic maturation has been discussed in several papers, but no formal process or algorithm has been proposed for large amounts of corrective actions in which faulty or false alarms could be occurring. In [7], the author points out there are unexpected and unplanned system interactions that can degrade the performance of the diagnostic design. In order to increase performance of the diagnostic model, historical maintenance actions will be used to help mature the model. However, [7] points out that



	M1	*M2*	*M3*
M1	0	1	1
M2	0	0	1
M3	0	0	0

Figure 4. The TFPG model and corresponding D-Matrix after the logical closure has been taken of the TFPG model in Figure 3.



	M1	*M2*	*M3*
A	1	0	0
B	0	1	0
C	0	0	1

	M1	*M2*	*M3*
M1	0	1	0
M2	0	0	1
M3	0	0	0

Figure 5. The resulting TFPG model and D-Matrices once the logical unclosure has been taken of the TFPG model in Figure 4. The bottom matrix shows the first order relations between the alarms and faults while the bottom matrix shows the relations between alarms.

the process requires ready access to the model, maintenance events, and any other information that could aid in the maturation process. In order to utilize all of these resources, the author proposes using an ontology design to gather all the required data together in a meaningful way.

In [17], [18], the authors discuss using explanation-based learning for the diagnostic model. If misdiagnosis occurs, then additional testing is done until a correct diagnosis has been made. This information can then be used to modify the structure of the model so that the correct diagnosis is consistent with testing. The authors also give a proof that given a single misdiagnoses, the model can be modified so that the misdiagnoses never occurs again. However, this was only valid for a single training example and did not include how to deal with faulty or false alarms.

In [20], the authors also use explanation based learning to aid rule-based diagnostics. The authors use fault diagnostic cases to help create heuristic domain knowledge that would then assist the reasoner. This heuristic domain knowledge was then used to create additional rules which would then be used in conjunction with the original rule based reasoner.

The authors in [9] also discuss the need for diagnostic maturation. In the paper, the authors discuss the need for recording flight information and maintenance data. They present an at-wing modular application for portable maintenance aids which can assist maintenance events by giving information to maintenance workers.

4. ONTOLOGY-GUIDED DATA MINING

In previous work, Wilmering and Sheppard suggested using domain ontologies as a means to focus and filter data analysis in data mining [10]. The specific focus of that work was utilizing the ontologies to guide the process by which diagnostic models could be matured over time. In this paper, we used domain ontologies as a way to join together different data sources and to find discrepancies between those different data sources.

The approach taken in developing ontologies to support the knowledge discovery process is based on a set of standardized semantic models developed in the EXPRESS modeling language [11], [12]. EXPRESS is an information modeling language defined by the International Organization for Standardization (ISO) to support communication of product data between engineering applications. The purpose of the language is to define the semantics of information that will be generated by a system and is not meant to define database formats, file formats, or exchange formats. In EXPRESS, models are defined using a hierarchy partitioned along schemata, entities, and attributes [19]. EXPRESS is often described as being object oriented in flavor because it incorporates a number of object-oriented features, such as encapsulation, abstraction, and inheritance. Values for attributes can be constrained through logical constraints on those attributes. These constraints, which often define relationships in non-trivial ways, give EXPRESS the ability to define computer-processable semantics. These constraints allow applications to discern if the information being received satisfies the intended meaning when it was generated and transmitted [19].

In this application, we used ontologies derived from the IEEE Std 1232 Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE) and IEEE Std 1636 Software Interface for Maintenance Information Collection and Analysis (SIMICA) [11]. [12]. Specifically, we used the IEEE Std 1232 (AI-ESTATE), and IEEE Std 1636.2 (Maintenance Action Information or MAI). AI-ESTATE is a set of specifications for exchanging data and defining software services for diagnostic systems. Its purpose is to standardize the interfaces between elements of an intelligent diagnostic reasoner as well as the representation of diagnostic knowledge and data for use by such diagnostic reasoners. The information models defined for AI-ESTATE are designed to form the basis for facilitating exchange of persistent diagnostic information between two reasoners, and also to provide a formal typing system for diagnostic services. The principal information model used out of AI-ESTATE for this work is the D-Matrix Inference Model (DIM) since it can be mapped to the structure of Timed Failure Propagation Graphs. An additional key information model—the Dynamic Context Model—also provides the semantics for historical information captured by a reasoner during a diagnostic session. Finally, both of these models make use of a "common" information model (called the Common Element Model) [11]. The SIMICA MAI was designed to capture records of actual maintenance actions performed on a particular system or subsystem [12].

Recent work in ontology-guided data mining has made use of standard ontology languages (e.g., OWL, DAML+OIL, and RDF) [14], [21], [22]. EXPRESS was not designed to support ontology-based analysis; however, the semantics defined by EXPRESS models are very rich. Therefore, we used the EXPRESS models as the foundation for defining ontologies using one of the most widely used ontology languages. We decided to use the Web Ontology Language (OWL) due to its prevalence in ontology-based systems.

The Web Ontology Language, or OWL, is a language for defining and instantiating ontologies [14]. An OWL ontology may have descriptions of classes, properties, and their instances. The formal OWL semantics then specify how to find logical consequences from the defined entities.

To convert EXPRESS to OWL, we first had to define a mapping of EXPRESS concepts to OWL concepts. Once the mapping of concepts was defined, we then created all of the OWL ontologies based on the EXPRESS standards. Finally, we converted all of the data into the OWL format and used our ontology guided data mining algorithm to locate the discrepancies.

The following code is part of the EXPRESS definition of the AI-ESTATE DIM and the EXPRESS diagram can be seen Figure 6 [11]. Part of the corresponding OWL definition for the AI-ESTATE DIM can be seen in Figure 7.

```
ENTITY OutcomeInference;
   andOrRows                : SET [1:?] OF Inference;
   preconditionTestOutcome : TestOutcome;
   confidence               : OPTIONAL ConfidenceValue;
   andOrRelation            : BOOLEAN;
 UNIQUE
   one_outcome              : preconditionTestOutcome;
 WHERE
   conjunctOrDisjunct : ((SELF.preconditionTestOutcome.valueDomain = Pass) AND
                (SELF.and_Or = TRUE)) XOR
                        ((SELF.preconditionTestOutcome.valueDomain = Fail) AND
                (SELF.and_Or = FALSE));
   noUserDefined      : reconditionTestOutcome.valueDomain[1] <> UserDefined;
END_ENTITY;
```

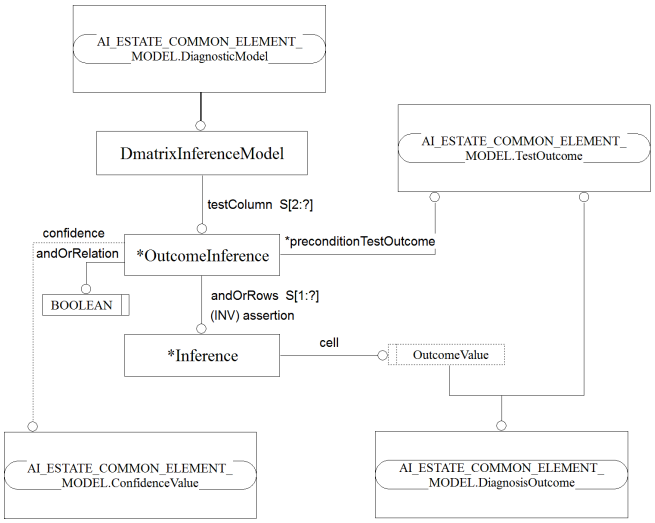


Figure 6. The EXPRESS code and diagram for the AI ESTATE D-Matrix Inference Model (DIM). The lines with circles and labels denote attributes while the lines with circles and no label denote subclass relationships.

For our mining algorithm, we located any discrepancies in our ontology where the reasoner's diagnosis and that of the maintenance event differed. Once those discrepancies were located, we pulled in and stored all of the alarm sequences corresponding to the easoner's wrong diagnosis. Those alarm sequences and the corresponding fault which was determined to be the true cause through the maintenance event are then used in the following section for the rest of the maturation algorithm.

5. Maturation of Alarm Dependencies

In TFPG models, alarms monitor or observe faults. If a certain alarm is monitoring a fault for a real world application, but maintenance events are finding that the alarm never occurs when that fault occurs, then that alarm probably should not be monitoring that particular fault. Additionally, if another alarm is not monitoring a fault but the alarm always occurs when the fault occurs, then that alarm should probably monitor that fault. In addition, there will also be alarms that do not fire when they should (non-detects) and alarms that fire when they should not (false alarms). These alarms need to be analyzed in order to gain an accurate picture of the alarms that should be occurring based on the maintenance events. Such analysis can also assist incorporating uncertainty measures into the diagnostic process.

In the maturation of alarm dependencies, we have a collection of alarm sequences from whenever maintenance find a certain fault. The maturation process will search for any differences between the alarm sequences and signatures in the D-matrix. However, care needs to be taken when considering false alarms and non-detects since adjusting the dependencies based on those alarms will likely have a negative effect on the performance of the reasoner.

```
<owl:Class rdf:ID="OutcomeInference">
    <rdfs:subClassOf rdf:about="#DMATRIX_MODEL"/>
</owl:Class>
<owl:Class rdf:ID="CEM_ConfidenceValue">
    <rdfs:subClassOf rdf:about="#DMATRIX_MODEL"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="confidence">
    <owl:maxCardinality  rdf:datatype=
        "http://…/XMLSchema#nonNegativeInteger">1
    </owl:maxCardinality>
    <owl:minCardinality rdf:datatype=
        "http://.../XMLSchema#nonNegativeInteger" >0
    </owl:minCardinality>
    <rdfs:domain rdf:resource="#OutcomeInference"/>
    <rdfs:range rdf:resource="#CEM_ConfidenceValue"/>
</owl:ObjectProperty>
```

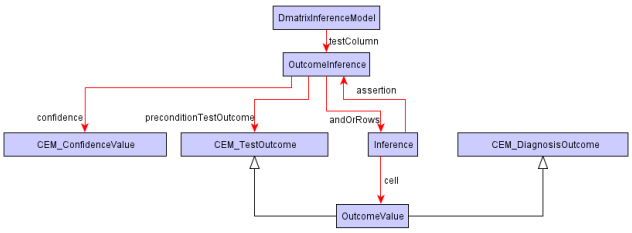


Figure 7. Part of the OWL code and diagram for the AI ESTATE DIM model. The large arrows without labels denote parent and child relationships while the smaller arrows denote relationships.

The maturation algorithm we developed works as follows. First, we retrieve the alarm sequences corresponding to the AI-ESTATE based and SIMICA MAI based logs of a repaired fault (whether the diagnosis was correct or not). An alarm sequence is represented as a bit string where each position in the bit string corresponds to a different observable alarm in the TFPG model. For example, a one at index *i* means that the *i*th alarm fired in the fault sequence, and a zero means that the *i*th alarm did not fire for that fault sequence. We then sum and normalize each bit yielding a probability of firing given the fault was diagnosed as the true fault. Finally, we compare the resulting vector of probabilities to the fault signature in the AI-ESTATE DIM-based D-matrix that corresponds to the repaired fault. Where

there is a wide disparity between the bit positions in the D-matrix and the probabilities in the probability vector, we flag that as a relationship to be examined. The following are the steps for the process:

1. Find all discrepancies between the maintenance diagnosis and the reasoner diagnosis.
2. From those discrepancies, pull in all of the alarm sequences for a particular maintenance diagnosis (or fault).
3. Calculate the probability of each alarm occurring given the maintenance diagnosis.
4. Compare each alarm probability with the D-matrix.
    a. If the alarm is occurring with a high probability, but the D-matrix does not have the alarm observing the fault, then flag that alarm to be looked at in relation to that fault.
    b. If the alarm is occurring with a low probability, but the D-matrix shows a relationship between the fault and alarm, then flag the alarm to be looked at in relation to the fault.

If an alarm occurs with a high probability and the D-matrix shows that the alarm is observing the fault or if the alarm occurs with a low probability and the D-matrix shows that the alarm is not observing the fault, then there appears to be no problems with that alarm with respect to the fault. However, in certain cases, it may still be beneficial to look at those alarms. For example, if an alarm is occurring with a probability of .35 and the D-matrix shows no relationship between that fault and alarm, then that could suggest that the alarm is faulty and needs to be analyzed more carefully.

6. Maturation of Alarm Sequences

This basic TFPG maturation described in the previous section is very simple and does not make use of the ontologies. In this section, we use the casual semantics of the TFPG and D-matrix to define an algorithm for maturing causal temporal relationships.

Similar to the maturation process of alarm dependencies, we will have a set of alarm sequences from whenever maintenance found a particular fault. The difference is that instead of just looking at which alarms were triggered, we also look at the order in which the alarms were triggered. We will still have false and non-detect alarms and special steps must be taken in dealing with these alarms.

Given a set of alarm sequences, we calculate a post-occurrence probability matrix which gives the probability of an alarm occurring after another alarm. For alarms *i* and *j*, [*i*,*j*] represents the probability that alarm *i* occurred before alarm *j* with respect to the total number of alarm sequences that have occurred. This matrix represents the temporal occurrences of the alarms that are being observed.

Next, we trace the expected alarm sequences for the TFPG model. In the simple case, one walks the unclosed D-Matrix which will give a straight sequence of alarms. However, in many cases there will be a split in the propagation link in the TFPG model. For example if fault 1 in Figure 1 were to fail, the signal would split and propagate to alarms M2 and M3. Because the propagation signal is split, alarms M2 and M3 could occur in either order and still result in the correct diagnosis. This makes trying to calculate the ideal alarm sequence difficult. One possible way would be to find all of the possible different valid alarm sequences for a given fault and then calculate a transition probability matrix for those expected alarms. The expected matrix could then be compared to the actual post-occurrence probability matrix to discover discrepancies. However, generating all of the valid alarm sequences is a combinatorial problem and therefore has an exponential run time. Therefore, we simply find for each alarm, which alarms we could expect to see following that alarm for a given fault. To do so, we start tracing an alarm sequence in the unclosed D-Matrix. If we find a split in the model, we follow both of those sequences and find the straight sequence of those alarms. Then we compare each alarm sequence. For each alarm in the alarm sequences that occurred after the split, we compare each alarm in one alarm sequence against all other alarms in the other alarm sequences such that those alarms are marked to follow the given alarm. For straight alarm sequences, we specially mark those alarms and the following alarms so that it is known which alarm has to occur first. For example, given the simple TFPG model in Figure 8, we see that the signal is split at the discrepancy D2.

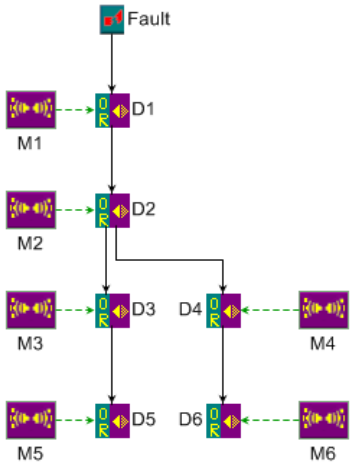


Figure 8. A simple TFPG model. The signal is split at the discrepancy D2 and makes predicting the expected alarm sequences a difficult problem.

We first start walking down the alarms starting at the fault and we generate the alarm sequence M1, M2. We then find a split occurring after alarm M2, generate all of the alarm sequences that occur after the split, which would give us the sequences M3, M5 and M4, M6. We first mark which alarms must occur before the other alarms, such as M1 must occur before all other alarms. Similarly, M4 must occur

before M6. Next, we compare the two alarm sequences that occurred after the split. In this case, M3 may occur before M4 and M6. Similarly, M4 may occur before M3 and M5. After these sequences of ordering notes have been made, we then compare them to the post-occurrence probability matrix and note any discrepancies we are seeing between the two for inspection. For the TFPG model in Figure 8, we would see the post-occurrence matrix.

	M1	M2	M3	M4	M5	M6
M1		X	X	X	X	X
M2			X	X	X	X
M3				O	X	O
M4			O		O	X
M5				O		O
M6			O		O	

The squares with X's denote observations that should always occur (that is, the probabilities should be close to 100 percent) while the squares with O's denote observations that may occur with high probability. The following steps summarize the algorithm:

1. Find all discrepancies between the maintenance diagnosis and the reasoned diagnosis.
2. From those discrepancies, pull in all of the alarm sequences for a particular maintenance diagnosis (or fault).
3. For each alarm sequence, calculate the transition probability matrix where [*i*,*j*] gives the probability of alarm *j* occurring after alarm *i*.
4. For the particular fault, mark the alarms that are expected to have high probability values.
5. Compare the marked matrix with the transition probability matrix and look for any discrepancies.
6. If there are any discrepancies between the two, flag the two alarms to be looked at.

Similarly to the alarm-fault maturation process, some probability values may not fall into a gray area. Those alarms may not be require the TFPG model to be modified, but may have other problems, such as being a bad alarm.

In addition to trying to predict alarm sequences, we also propose comparing these temporal matrices to the logical unclosed D-matrix and the adjacency matrix of the TFPG model. This allows us to see the relationship between the temporal, logical, and actual relationships of the alarms with one another in the TFPG model. In comparing these three matrices, we propose taking a specific corrective action if there are discrepancies between these three matrices which will correct any errors that may be present in the TFPG model. This aspect of the algorithm is still being developed.

7. EXPERIMENTS

To test our fault dependency maturation algorithm, we used a Pump and Valve TFPG Model developed by Boeing and used the Vanderbilt FACT diagnostic reasoner to diagnosis faults given a firing of alarms. The Pump and Valve model provided was used as "ground truth," and two erroneous models were created by deleting a link and removing a relationship in the TFPG and adding a link and thus creating a relationship between a fault and an alarm in the TFPG model that should not exist. Three sets of alarm sequences were used, one of which properly identified the P01_burst fault in the ground truth TFPG. The other two, while not capable of fully isolating P01_burst still yields hypotheses consistent with this fault. The four alarms correctly identify P01_burst as the fault are the following:

- IVHM09 – In flight operating pressure command Low output Lo
- IVHM11– In flight operating pressure command Medium, output Lo
- IVHM13– In flight operating pressure command High, output Lo
- IVHM15– Fuel Containment

In our first modified model, we deleted the relationship between P01_burst and IVHM15 through deleting the link from P01_burst to the discrepancy Contain_Fuel_in_Plumbing Failed. The good model can be seen in Figure 9 and the bad or modified model can be seen in Figure 10.

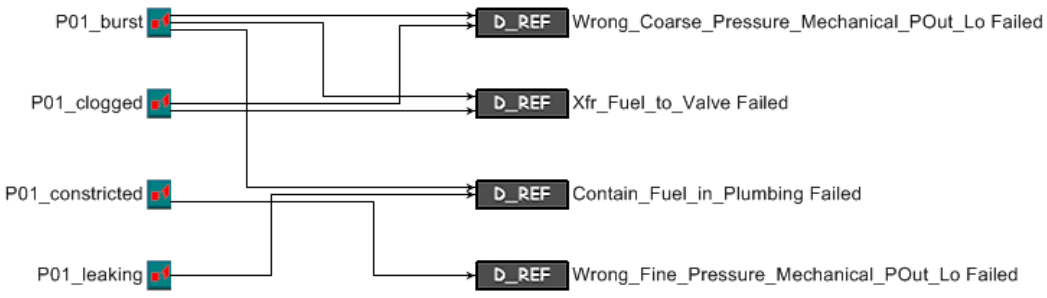


Figure 9. Part of the good Pump and Valve TFPG model. The model was used to diagnosis the ground truth and to simulate maintenance events that would discover the true fault in the bad model.

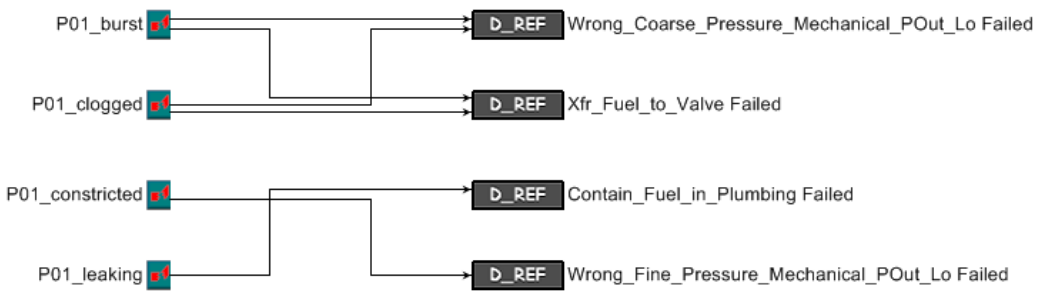


Figure 10. Part of the bad Pump and Valve TFPG model. Note that a link has been removed from P01_bust to the discrepancy Contain_Fuel_in_Plumbing Failed.

First, we fired alarms IVHM09, IVHM11, and IVHM13. Next we only fired IVHM15. Finally, we fired the alarms IVHM09, IVHM11, IVHM13 and IVHM15. In addition, we added false alarms IVHM01, IVHM04, and IVHM05 where each individual false alarm was fired somewhere in the alarm sequence of IVHM09, IVHM11, IVHM13 and IVHM15. From these seven alarm sequences, we then repeated the sequences multiple times to simulate reoccurrence of an alarm sequence in a real world application. The alarm sequences that contained false alarms were repeated fewer times than the alarm sequences that did not contain any false alarms.

As stated above, the alarms IVHM09, IVHM11, IVHM13 and IVHM15 properly diagnoses the P01_burst fault in the ground truth model. However, since we deleted the causal link from P01_burst to IVHM09 in the modified model, the reasoner diagnosed different faults. Specifically, the reasoner found that there were 50 different possible faults that could have occurred when the alarms were IVHM09, IVHM11, IVHM13 and IVHM15 fired, none of which included P01_burst. To finish the test case, we assumed that maintenance would eventually determine that P01_burst was the actual fault so that the system could be repaired. This information would be provided by the maintenance action data.

Using these alarm sequences, we applied the algorithm described above to see if we could determine where the correct dependency in the model should be. In using the TFPG maturation algorithm, we created the following table as output, which can be seen in Figure 11. The table gives the corresponding fault for which we are investigating in the first row. The "Index" column gives the index of the alarms while the "Alarms" column lists every alarm in the TFPG model. The "DMatrix" column shows if the alarm observes the particular fault. A 1 means the alarm does observe the fault while a 0 means the alarm does not observe the fault. The "FaultSequence" column gives the probability of the alarm occurring given that the fault was diagnosed as the fault through the maintenance event.

After analyzing the output, we found that the algorithm identified that the TFPG model did not have the alarms IVHM15 monitoring the fault P01_burst but noticed that the alarm IVHM15 occurred with a high probability if the fault P01_burst was diagnosed as the true fault in the maintenance event and thus suggests that some relationship exists between P01_bust and IVHM15. In our table, we have flagged that alarm to be analyzed to see if there is an error in the model. After examining the TFPG model, we found that links from the fault P01_burst to alarms IVHM15 should be added. In our case, the simplest explanation was to add a link from P01_burst to the discrepancy Contain_Fuel_in_Plumbing Failed. As indicated above, this was indeed the link that was deleted from the original model.

P01_burst			
Index	Alarms	DMatrix	FaultSequence
1	LBIT13	0	0
2	LBIT14	0	0
3	IVHM01	0	0.118
4	IVHM02	0	0
5	IVHM03	0	0
6	IVHM04	0	0.049
7	IVHM05	0	0.079
8	IVHM06	0	0
9	IVHM07	0	0
10	IVHM08	0	0
11	IVHM09	1	0.643
12	IVHM10	0	0
13	IVHM11	1	0.643
14	IVHM12	0	0
15	IVHM13	1	0.643
16	IVHM14	0	0
17	IVHM15	0	0.722
18	OFBD01	0	0
19	OFBD02	0	0
20	OFBD03	0	0

Figure 11. Part of the output generated by our algorithm. The alarm IVHM15 is not observing the fault P01_bust, but is occurring with a large number of times whenever the fault P01_bust is being diagnosed as the true fault, leading to suggest that the alarm IVHM15 should observe the fault P01_burst.

We also tested a scenario in which we added an extra link into the TFPG model. This was done by adding a relationship between P01_burst and IVHM05 through adding a link between the discrepancy PV_1C_Transfer_Fuel_to_Engine Failed and IVHM05. Similar to the previous demo, we used the same set of

alarms on our bad model and we assumed that maintenance would eventually determine that P01_burst was the actual fault. The sample output for the scenario can be seen in Figure 12.

P01_burst			
Index	Alarms	DMatrix	FaultSequence
1	LBIT13	0	0
2	LBIT14	0	0
3	IVHM01	0	0.118
4	IVHM02	0	0
5	IVHM03	0	0
6	IVHM04	0	0.049
7	IVHM05	1	0.079
8	IVHM06	0	0
9	IVHM07	0	0
10	IVHM08	0	0
11	IVHM09	1	0.643
12	IVHM10	0	0
13	IVHM11	1	0.643
14	IVHM12	0	0
15	IVHM13	1	0.643
16	IVHM14	0	0
17	IVHM15	1	0.722
18	OFBD01	0	0
19	OFBD02	0	0
20	OFBD03	0	0

Figure 8. Part of the output generated by our TFPG maturation algorithm. The alarm IVHM05 is set to observe the fault P01_bust, but is occurs a low number of times whenever the fault P01_bust is being diagnosed as the true fault. This suggests that IVHM05 should not be observing the fault P01_burst.

8. Conclusion

We hypothesized that the alarm dependency maturation algorithm would be able to find the missing link in the faulty TFPG model. In addition, the subtraction of a link was only for a specific test case. More experimentation is needed in which a large variety of links is added or deleted to fully test whether our algorithm can find all missing or added links. Once we are able to run the algorithm on a variety of scenarios, we will have a better picture of how well the algorithm performs. In addition, while we've developed an algorithm that will mature how the alarms and discrepancies relate to each other, we have yet to fully test the algorithm.

9. Future Work

The initial experimentation of this algorithm is promising. It was able to find the deleted dependencies in the TFPG model. However, these were simple test cases and more work is still needed to test if it is generally able to find the added or deleted dependency in the TFPG model. In addition, the alarm sequence maturation algorithm needs to be tested. As already stated above, the algorithm at the time of writing was developed but not fully tested.

We would also like to further develop our maturation algorithms. Currently, we are dealing with just the alarm sequences that occurred with the reasoner misdiagnosed the fault. However, just because an alarm sequence resulted in a correct diagnosis does not mean that the model is correct. For example, it could be that all of the other possible faults are just more unlikely. Therefore, we want to compare fault sequences of both negative and positive diagnoses. Given these two groups of alarm sequences, we could then compare the probabilities of alarms occurring and compare those probabilities between the two groups. Furthermore, these probabilities could then be compared to the D-matrix. Another possible way to further develop the algorithms may be to use some form of hierarchical clustering algorithm.

Other future work will also include maturating the time intervals used on the TFPG models. Since the diagnostic algorithms use the time intervals to diagnosis a fault, a wrong time value could greatly change how the reasoner diagnoses a fault. Again, using an alarm sequence from a maintenance event, one could be able to find those faulty time values and adjust them.

Finally, we would like to include maturation of probabilistic values in the TFPG models that utilize probabilistic values. If those probabilities are faulty, then the diagnostic reasoner could end up diagnosing the wrong faults. Again, if we have the maintenance event which informs us which alarms were triggered and what fault was actually found during maintenance, we could find those faulty values in the model and recommend changes to them.

Acknowledgments

We would like to thank The Boeing Company for providing us with the Pump and Valve model and assistance in understanding and using the model. Also, we would like to thank Vanderbilt University for their FACT software and GME modeling tool and helping us use the software. This research was funded in part by a grant from NASA under the aviation safety program.

References

[1] A. Misra, *Sensor-based diagnosis of dynamical systems*, Ph.D. dissertation, Nashville, TN, USA, 1994.

[2] A. Misra, J. Sztipanovitz, and J. R. Carnes, "Robust diagnostic system: structural redundancy approach," W. Buntine and D. H. Fisher, Eds., vol. 2244, no. 1. SPIE, 1994, pp. 249–260.

[3] S. Abdelwahed, G. Karsai, and G. Biswas, "A consistency-based robust diagnosis approach for temporal causal systems," *Proceedings of the 16th International Workshop on Principles of Diagnosis*, 2005, pp. 73–79.

[4] S. Abdelwahed, G. Karsai, N. Mahadevan, and S. Ofsthun, "Practical implementation of diagnosis systems using timed failure propagation graph models," *IEEE Transactions* on *Instrumentation and Measurement,* vol. 58, no. 2, pp. 240 –247, February. 2009.

[5] S. Abdelwahed, G. Karsai, and G. Biswas, "System diagnosis using hybrid failure propagation graphs," *Proceedings* of t*he 15th International Workshop on Principles of Diagnosis*, 2004.

[6] N. Mahadevan, S. Abdelwahed, A. Dubey, G. Karsai, "Distributed diagnosis of complex systems using timed failure propagation graph models," *IEEE AUTOTESTCON 2010 Conference Record*, pp. 124-129, September 13-16, 2010.

[7] T. J. Wilmering, "Semantic requirements on information integration for diagnostic maturation," *IEEE AUTOTESTCON 2001 Conference Record*, pp.793-807, 2001.

[8] T.J. Wilmering, "When good diagnostics go bad - Why maturation is still hard," *Proceedings of the IEEE 2003 Aerospace Conference, 2003*. vol.7, pp. 3137- 3147, March 8-15, 2003.

[9] C.S. Byington, P.W. Kalgren, and B.P. Donovan, "Portable diagnostic reasoning for improved avionics maintenance and information capture & continuity," *IEEE AUTOTESTCON 2004 Conference Record.* pp. 518- 524, September 22-23 2004.

[10] Timothy J. Wilmering and John W. Sheppard, "Ontologies for Data Mining and Knowledge Discovery to Support Diagnostic Maturation," *Proceedings of the 18th International Workshop on Principles of Diagnosis* (DX-07), Nashville, TN, May 2007, pp. 210-217.

[11] IEEE Std 1232-2011, *IEEE Standard for Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE)*. IEEE Standards Press, Piscataway, New Jersey, 2011.

[12] IEEE Std 1636.2-2011, *IEEE Trial-Use Standard for Software Interface for Maintenance Information Collection and Analysis (SIMICA): Maintenance Action Information (MAI)*. IEEE Standards Associated Press, Piscataway, New Jersey 2010.

[13] ISO 10303-11:2004, *Industrial automation systems and integration -- Product data representation and exchange -- Part 11: Description methods: The EXPRESS language reference manual.* International Organization for Standardization, Geneva, Switzerland.

[14] *OWL 2 Web Ontology Language Document Overview*. W3C. 2009-10-27. http://www.w3.org/TR/owl2-overview/.

[15] J. W. Sheppard and S. G. Butcher. 2007. "A Formal Analysis of Fault Diagnosis with D-matrices". *J. Electron. Test.* 23, pp.309-322. August 2007,

[16] Scott Wahl, "Logical Closure for Diagnostic Network Simplification," NISL Technical Report, Department of Computer Science, Montana State University, May 2009.

[17] W. Simpson and J. Sheppard, *System test and diagnosis*. Kluwer Academic Publishers Norwell, Massachusetts, 1994.

[18] J.W. Sheppard, "Explanation-based learning with diagnostic models," *IEEE AUTOTESTCON 1992 Conference Record*, pp.159-166, September 21-24, 1992.

[19] Mark A. Kaufman, John W. Sheppard, and Timothy J. Wilmering, "Model-Based Standards for Diagnostic and Maintenance Information Integration," *IEEE AUTOTESTCON 2007 Conference Record*, Baltimore, MD, September 2007, pp. 304-310.

[20] S. Kobayashi and K. Nakamura, "Knowledge compilation and refinement for fault diagnosis," *IEEE Expert*, vol. 6, no. 5, pp. 39 –46, October 1991.

[21] DAML+OIL. Joint US/EU ad hoc Agent Markup Language Committee. 2001-3-27. http://www.daml.org/2001/03/daml+oil-index.

[22] Resource Description Framework (RDF). 2004-02-10. W3C. http://www.w3.org/RDF/.

BIOGRAPHY

Shane Strasser *is currently pursuing his Masters in route to a PhD in computer science at Montana State University. He previously received a BS in computer science and mathematics from the University of Sioux Falls in Sioux Falls, South Dakota. His research interests are primarily in artificial intelligence and machine learning with a focus on prognostic of health management systems.*

John Sheppard *is the RightNow Technologies Distinguished Professor of Computer Science at Montana State University. He is also an Associate Research Professor at Johns Hopkins University. His research interests include algorithms for diagnostic and prognostic reasoning, machine learning and data mining in temporal systems, and reinforcement learning. Dr. Sheppard holds a BS in computer science from Southern Methodist University and an MS and PhD in computer science from Johns Hopkins University. He is a Fellow of the IEEE and currently serves as Co-Chair of the Diagnostic and Maintenance Control subcommittee of IEEE Standards Coordinating Committee 20 (SCC20) on Test and Diagnosis for Electronic Systems.*

Michael Schuh *is currently pursuing his PhD in computer science at Montana State University. He previously received a BS in computer science with math and business minors from the University of Wisconsin Oshkosh. His primary research interests are data mining and machine learning with a focus on large scale data and the Web.*

Rafal Angryk *is an associate Professor in the Computer Science Department at Montana State University. He received his M.S. and Ph.D. in computer science degree from Tulane University, New Orleans. His current research interests lie in the areas of Data Mining, Databases (Spatial Databases, Fuzzy Database Models), Distributed Systems (Mobile Agents Technology, Distributed Databases), and Artificial Intelligence (Fuzzy Expert Systems, Neural Networks).*

Clemente Izurieta *is an associate research professor in the Computer Science department at Montana State University. Born in Santiago, Chile, his research interests include empirical software engineering, design and architecture of software systems, design patterns, the measurement of software quality and ecological modeling. Dr. Izurieta has approximately 16 years experience working for various R&D labs at Hewlett Packard and Intel Corporation.*