Standardizing Diagnostic Models for System Test and Diagnosis

John W. Sheppard
ARINC Research Corporation
2551 Riva Road
Annapolis, MD 21401
Phone (410) 266-2099, FAX (410) 266-4010, E-mail: sheppard@arinc.com

Abstract: **Recent standardization activities by the IEEE are focusing on defining test specification languages, services between components of test environments, and most recently, interfaces and interchange formats for intelligent test systems. Currently, the Artificial Intelligence and Expert System Tie to Automatic Test Equipment (AI-ESTATE) set of standards (P1232) is defining formats and services for exchanging diagnostic knowledge between reasoners on next generation test systems. In this paper, we will describe the theoretical underpinnings of the first knowledge format being standardized by the Data and Knowledge Representation Working Group of AI-ESTATE. We will also provide a progress report on the standards activity.**

II. INTRODUCTION

Recent initiatives by the IEEE and the Department of Defense on standardizing automatic test equipment architectures have provided a unique opportunity to improve the development of test systems. The "A Broad Based Environment for Test" (ABBET™) initiative (IEEE PAR 1226) is attempting to provide the next generation in ATE and TPS development by standardizing test services and test development tools [1]. By using standardized methodologies in developing test executives, test programs, and communication protocols, ABBET conformant test systems will be interoperable, have transportable software, and move beyond vendor/product specific test stations. Further, the "Artificial Intelligence and Expert System Tie to Automatic Test Equipment" (AI-ESTATE) initiative (IEEE PAR 1232) will provide architectures for standardizing the test services of reasoning systems (including traditional fault trees, expert systems, model based systems, simulation based systems, and neural networks) and provide guidance on the test strategy services for the ABBET architecture [2].

Central to the ABBET/AI-ESTATE initiatives is the desire to incorporate the object oriented paradigm in specifying test resources, test programs, and test services. The various component standards of ABBET and AI-ESTATE are using information modeling based on the ISO Express object oriented modeling language. In this paper we will focus on one of the standards proposed under AI-ESTATE for developing diagnostic models. This particular standard proposes models for the object oriented design of fault trees and enhanced diagnostic inference models (EDIM). In addition, the proposed standard includes specifications of attribute models and common element models. These correspond to specific classes of objects that would be specified for a particular test architecture. The focus of the paper will be on the theoretical basis for the EDIM and its use in the industry.

The ISO Express models proposed for AI-ESTATE provide a standard representation of the common data elements required for test strategy optimization at the system level. Together with other standards within AI-ESTATE and ABBET, this will insure portability of test strategy related knowledge bases for intelligent equipment test and diagnosis. The specific goals of the component standard include incorporating domain specific terminology, providing extensibility, insuring portability of test executives and reasoning systems, enforcing interoperability of test resources and test equipment, providing traceability of the test process and the test development process, and permitting object oriented testing through the use of test encapsulation.

II. AI-ESTATE

The "Artificial Intelligence and Expert System Tie to ATE" (AI-ESTATE) standard P1232 is being developed to standardize on the interfaces between test systems and artificial intelligence based systems. In addition, AI-ESTATE is including standard representations for several types of knowledge bases and databases. Currently, the standard specifies representations for fault tree models (FTMs) and enhanced diagnostic inference models (EDIMs).

Currently, AI-ESTATE is being developed using a cooperative processing model [3]. Under this model, all processes communicate across a communications pathway or bus and access other parts of the system through a set of services.

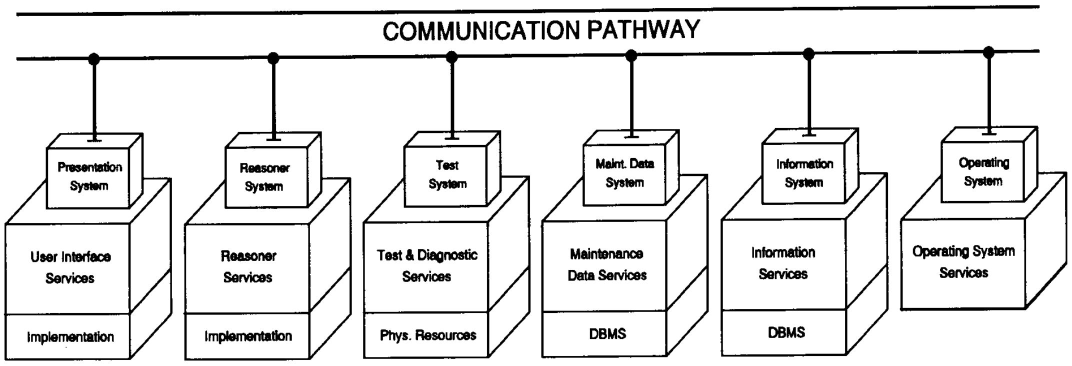

Fig 1. Architecture for an AI-ESTATE test system.

Specifically, each functional block is defined by the operations that can be performed on or by that block. The attributes of the block are specified in a class lattice in order to maximize reuse of components. If one functional block needs to interact with another, it does so by using the services provided by that block. Using this architectural concept, the objects communicating in an AI-ESTATE system include the test system, the reasoner, the human presentation system, a maintenance data collection system, the unit under test, and the operating system. This architectural concept is depicted in Fig. 1. Any data that is used by the objects on the pathway will be specified in some standard representation. Even though we currently have specifications for only the FTM and the EDIM, we anticipate defining or referencing standards for maintenance data collection databases and other databases and knowledge bases.

The current model representations defined by the AI-ESTATE P1232.1 Data and Knowledge Specification are specified using the Express modeling language. Currently, AI-ESTATE has specified four models including a common element model (a collection of entities used by all models), an attribute model (a collection of attributes inherited by several of the models), a fault tree model (a standard representation for a static fault tree), and the Enhanced Diagnostic Inference Model (a standard representation that extends the concepts of the dependency model and the information flow model as discussed below).

The structure of a fault tree can be viewed as a decision tree or table [4]. Each node of the tree corresponds to a test with some set of outcomes. The outcomes of the tests are branches extending from the test node to other tests or fault isolation conclusions. Typically, TPSs are designed around static fault trees; therefore, the AI-ESTATE subcommittee decided it was prudent to include a representation for the fault tree in its standard even though fault trees are not typically considered AI systems. The fault tree will serve as a minimal representation for test related data.

The AI-ESTATE fault tree model inherits elements and attributes of the common element model and the attribute model. The fault tree is processed by starting at the first test step, executing the indicated test, and traversing the branch corresponding to the test's outcome. The procedure is followed recursively until a leaf is reached in the tree indicating a fault isolation has occurred.

The EDIM is based on concepts of dependency modeling [5] and information flow modeling [6]. As with the FTM, the EDIM inherits elements and attributes from the common element model and the attribute model. In the EDIM, diagnostic test outcomes have been generalized beyond pass and fail outcomes to multiple outcomes for a diagnostic test. Inferences are identified between particular test outcomes and other test outcomes and diagnostic conclusions.

The set of inferences associated with a particular test outcome are represented in disjunctive normal form. This representation provides flexibility and consistency in the logical expression of the inferences. No negations are permitted within the AI-ESTATE representation since negative terms are expressly specified as atoms in the conjunctions. In other words, each conjunction consists of a set of positive and negative inference atoms, and the negative inference atoms are assumed to be negated.

The EDIM can be used by an AI system in several ways. The most basic approach is to limit test outcomes to pass and fail and to assume the test outcomes are symmetric. This leads to a traditional dependency model as used by several diagnostic tools in the industry. By allowing multiple failure inference, various grouping operations, asymmetric inference, and reasoning under uncertainty, the model extends to the information flow model as described in ref. 6. Finally, the EDIM extends further by permitting more detailed specification of test and UUT data, multiple test outcomes, and an extension mechanism (called the EXTEND schema) similar to the EXTEND verb of ATLAS [7]. The EXTEND schema provides tremendous expressive power while still controlling conformance to the standard. For example, using the EXTEND schema, causal relationships between fault isolation conclusions can be specified, thus extending the EDIM to include a causal model. For example, the EXTEND schema can also be used to include constraint information in case constraint satisfaction is used for diagnosis. Thus the EDIM has great power to provide many AI based solutions.

An important notion in developing object oriented test systems that include AI for test sequencing and inference is the ability to isolate a test from the remainder of the test set. This test isolation is referred to as *test encapsulation* [8]. An encapsulated test is an atomic test element defined to be independent of the current state of the UUT and the tester. In order to define an encapsulated test, preconditions of the tests and postconditions of the test need to be defined. The preconditions indicate the steps necessary to set up the test for execution, and the postconditions indicate the steps necessary to return the state to neutral. Individual tests and groups of tests can be encapsulated depending on the specific requirements of the tester and the UUT.

III. DIAGNOSTIC MODELING

The primary model under development in the current draft of P1232.1 is called the *Enhanced Diagnostic Inference Model* (EDIM). The EDIM has its basis in the dependency model of the 1980s. Historically, the dependency model was used to map relationships between functional entities in a system under test and tests that determine whether or not these functions are indeed being performed [9]. Typically, the model characterized the connectivity of the system under test from a functional perspective using observation points (or test points) as the junctions joining the functional entities together. If a portion of the system fed a test point, then it was claimed that the test associated with the test point *depended* on the function defined by that part of the system. This type of model is also referred to as a causal model [10], [11].

More recently, researchers and practitioners of diagnostic modeling found that this approach to modeling was problematic and could lead to inaccurate models. Thus, models developed under the functional or connectivity paradigm often led to inaccurate diagnostics. At this point, believing the algorithms processing the models were correct, work was done to identify the problems with the modeling approach and determine how to capitalize on the power of the algorithms processing these models without inventing a new approach to model-based diagnosis.

It was found that the focus of the model should be on the tests rather than on the functions or failures that may exist in the system [12]. In particular, the focus shifted from modeling functional or physical connectivity to relating inferences that can be drawn from real tests and their respective outcomes. From this focus, it was found that the dependency model was actually a representation of *logical* relationships between *test information* and *diagnoses* in the system under test. Such a representation can draw directly on principles from first order logic, and indeed the underlying algorithms processing these models do just that. It was at this point that the community began to recognize that the name *dependency model* was inappropriate.

In one of the first papers to address the issue of defining a diagnostic model, a new name was proposed for the model–the *information flow model* (IFM) [6]. The emphasis intended by using this name was to focus on the information provided by tests and how this information propagates via logical inference through a network of information sources in a diagnostic model.

An information flow model has two primitive elements: *tests* and *fault-isolation conclusions*. Tests include any source of information that can be used to determine the health state of a system. Fault isolation conclusions include failures of functionality, specific non-hardware failures (such as bus timing), specific multiple failures, and the absence of a failure indication. The information obtained may be a consequence of the system operation or a response to a test stimulus. Thus, we include observable symptoms of failure processes in the information flow model as tests. Doing this allows us to analyze situations that involve information sources other than formally defined tests. The purpose of the model, of course, is to combine these information sources (tests) to derive conclusions about the system being diagnosed.

After specifying the primitive elements of the model, the next step is to determine the logical relationships among the tests and between the tests and the conclusions (mostly the latter). The basic representation of the information flow model is a logical representation of the system being analyzed. In this representation, we define logical values for tests and fault-isolation conclusions. Specifically, if a test fails, it has a logic value of *true*; if a test passes, it is *false*. An asserted conclusion is *true*; a conclusion eliminated from consideration is *false*. To determine the logical relationships, an analyst will consider the following for each test:

- What inferences can be drawn from observing a test failing?
- What inferences can be drawn from observing a test passing?

In the initial stages of modeling, the first issue is more important. The modeler is interested in listing conclusions that, corresponding to a failure, would explain the considered test failure. The modeler is also interested in listing tests that, should they fail, would cause the current test to fail. If such tests do exist, we say that the current test *observes* tests that detect failure and conclusions that may fail. The second question is important in determining the type of test (e.g., whether the information provided is symmetric).

IV. THE DIAGNOSTIC INFERENCE MODEL

Several commercial and military tools process models based on the dependency model and the information flow model. One of these models is the *Weapon System Testability Analyzer* (WSTA) of the Integrated Diagnostic Support System (IDSS) owned by the U.S. Navy. Currently, the Navy is producing a handbook describing how to develop models fitting the description of section 3. Recognizing the problems associated with calling these models "dependency models," the Navy has also opted to change the name of their model, and they chose the name *Diagnostic Inference Model.* Again, the emphasis intended behind the name change was the relationship of information provided by tests and the inferences about diagnosis that can be drawn from specific test outcomes.

The DIM is an abstract representation of the information yielded by a unit under test when subjected to a well defined set of tests. This model is used to predict various system maintenance characteristics and derive optimum diagnostic test sequences that minimize maintenance costs. The DIM contains inference elements and maintenance specific elements. Tests and conclusions represent the inference elements of the DIM.

A test is an inference element of the DIM. It establishes a physical fact about the operation of a UUT from which UUT failure conclusions and other test conclusions are inferred. For the DIM, only two facts or outcomes are established by a test–*pass* and *fail.* The conclusions that can be inferred from these test outcomes cannot be determined until three physical characteristics of the test are defined–stimuli provided to the UUT, parameters to be observed, and pass/fail criteria.

In most physical systems, there is a range of values for observed parameters that are considered to be within an acceptable tolerance window for normal operation of the UUT. The range of values are used to establish the outcomes to be associated with performance of a test. Variations in real UUT component parameter values that remain within specified component tolerances are responsible for the range of expected values (as opposed to a single value) that are within the test limits. The range of values outside of the test limits are associated with UUTs containing failure modes that have a cause/effect relationship with the observed parameter.

The conclusion is a diagnostic element of the DIM. The conclusion represents the inference that can be drawn from a test outcome–that a component failure exists or does not exist and/or that another test should pass or should fail. DIM conclusions are assigned to test outcomes as inferences during model preparation. The conclusions are then processed by various testability algorithms to close the inference model and to evaluate UUT testability characteristics and compute optimized diagnostic test sequences.

The conclusion also can assume one of two values–*true* and *false.* A true value indicates that the associated component or input parameter is healthy according to the criteria of the conclusion. A false value indicates that the associated component or input parameter is faulty (note that this is opposite to the truth value assignments in the IFM). Inferred conclusion can be demonstrated by inserting faults into the system and observing test outcomes. A system contains a finite number of output parameters, component parameters and input parameters. Therefore, there are a finite number of conclusions for a system (one for each component parameter and input parameter and associated evaluation criteria). Combinations of conclusions can be incorporated directly or handled by the diagnostic algorithm of the tool used.

In the context of the DIM, a component failure mode is defined as a component parameter value that fails the evaluation criteria established in the associated conclusion. For a properly defined conclusion, the system no longer performs one or more of its functions in accordance with specified requirements when the component parameter associated with the conclusion fails the evaluation criteria. Therefore, a conclusion value of false establishes the existence of a component failure mode. In a well designed system, the component parameter value must fall outside the specified tolerance range for the associated component failure mode to exist. However, some systems have provision for alignment to bring a system up to specified performance when a parameter is not out of tolerance but the cumulative effect of all tolerance build-up is failure. Like component failures, alignment is treated as a conclusion within the DIM.

Components often have multiple failure modes. For example, a wirewound resistor may fail open, shorted, out of tolerance low resistance (but not completely shorted). Because each system output parameter can be expressed as a function of the input and component parameters and the sensitivity of each system output parameter to changes in a component value can vary markedly, system outputs may respond differently to various failure modes of the same component. Furthermore, if there are multiple failure modes related to a single component, the failure modes may be able to exist independently, each with its own effect on overall system performance. A conclusion must be prepared for each component and input failure mode.

V. THE EDIM

The AI-ESTATE subcommittee of SCC20 is working to standardize a neutral interchange format for models based on the information flow model or the diagnostic inference model. Because they recognize the modeling being standardized is an extension of the IFM and DIM; therefore, the committee has opted to name their model the *enhanced* DIM (EDIM). The purpose of this paper is not to provide the specification of the model since this can be found in the standard itself. Instead, we relate the EDIM to the DIM by pointing out the relevant parts of the standard and some of the enhancements.

In addition to the EDIM, AI-ESTATE defines two additional models intended to be references by all models specified in P1232.1. First, AI-ESTATE defines a Common Element Model which includes entities common to the domain of equipment test and diagnosis. They are intended to serve as the basic entities to be used by other model specifications in this document. The Common Element Model include the following entities.

```
ENTITY diagnosis;
    name:                   string;
    description:            optional string;
    corresponds_to:         system_aspect;
    repairable_at:          optional repair_level
    failure_information:    optional failure_probability;
    members:                set [0:?] of diagnosis;
END_ENTITY;
```

Thus diagnosis is a part-whole hierarchy consisting of additional diagnoses. These diagnoses are referred to as diagnostic units in the DIM and have been called by other names including aspects, components, fault isolation conclusions, or failure modes (depending on the model). Because the entity is defined to be a hierarchy, it also encompasses concepts such as line replaceable unit, line replaceable module, shop replaceable unit, weapon replaceable assembly, shop replaceable assembly, subsystem, etc.

Two additional attributes of the entity diagnosis should be noted. First, diagnosis "corresponds to" a system_aspect. The system_aspect is intended to represent a real physical part of the item tested so that appropriate repair actions can be identified. The system_aspect is defined by the following entity.

```
ENTITY system_aspect;
    name:                   string;
    description:            optional string;
    failure_information:    optional failure_rate;
    repaired_using:         repair_action;
    members:                set [0:?] of system_aspect;
END_ENTITY;
```

The second attribute of the entity diagnosis is the repair_level. This relates to the repair action attribute of the entity system_aspect. The repair_level describes the level in a maintenance hierarchy that is appropriate for repairing the fault corresponding to the system aspect of the diagnosis. The repair_action is the set of actions to be taken on that system_aspect. These two attributes are described with the following entity definitions.

```
ENTITY repair_level;
    name:                   string;
    description:            optional string;
    next_lower_level:       set [0:?] of repair_level;
END_ENTITY;

ENTITY repair_action;
    name:                   string;
    description:            optional string;
    requires_resources:     set [1:?] of resource;
    requires_parts:         set [0:?] of part;
    applicable_at:          repair_level;
    costs:                  set [0:?] of cost_attribute;
END_ENTITY;
```

Central to the EDIM (as well as the DIM) is the concept of a test. Therefore, we can expect a definition of a test to be included in the model specification. Because of the fundamental importance of the test in any diagnostic process, the Common Element Model includes the test definition.

```
ENTITY test;
    name:                   string;
    description:            optional string;
    costs:                  set [0:?] of cost_attribute;
    requires_resources:     set [1:?] of resource;
    has_outcome:            select (pass, fail);
    members:                set [0:?] of test;
END_ENTITY;
```

Once again, the test entity is a part-whole hierarchy where a given test may consist of a collection of several simpler tests. This hierarchy provides the ability to group tests together and infer information from a value assigned to the whole group (as well as individuals within the group). Since the DIM does not include the concept of a test inference from a group of tests, this notes the first enhancement of the EDIM over the DIM. We also point out that the EDIM extends the has_outcome attribute to be arbitrary. Also, a confidence can be associated with an outcome in the EDIM. The DIM restricts it to be one of a set of two possible outcomes corresponding to either pass or fail with an assumed confidence of 1.0.

We have seen in both the entity repair_action and test that resources are required to perform the necessary actions. Specific resources are not generally considered a part of the DIM, but we recommend specifying them to track the correlation of resources to model elements. The definition of the entity is very simple:

```
ENTITY resource;
    name:               string;
    description:        optional string;
    costs:              set [1:?] of cost_attribute;
END_ENTITY;
```

In addition to the Common Element Model, AI-ESTATE includes a specification of an Attribute Model. The AI-ESTATE Attribute Model lists several specific attributes that can be associated with tests, diagnoses, and resources. Specifically, the AI-ESTATE Attribute Model includes the following types of attributes: cost attributes, cost bounds, and failure rate.

The Diagnostic Inference Model is more than just a definition of tests, diagnoses, and cost attributes. Central to the DIM is the definition of inferences that are possible between tests and diagnoses. This is true of the EDIM as well. AI-ESTATE defines the EDIM as follows:

```
SCHEMA Enhanced_Diagnostic_Inference_Model;
REFERENCE FROM AI_ESTATE_Common_Element_Model(diagnosis, test,
        outcome, system_aspect, repair_level);
REFERENCE FROM AI_ESTATE_Attribute_Model (cost attributes);

ENTITY Enhanced_Diagnostic_Inference_Model;
    diagnostic_model:       set [1:?] of diagnosis;
    test_model:             set [1:?] of test;
    system_model:           set [1:?] of system_aspect;
    repair_structure:       set [1:?] of repair_level;
    outcome_model:          set [1:?] of outcome_infers;
    infer_structure:        set [0:?] of related_inference_group;
END_ENTITY;
END_SCHEMA;
```

The first four structures are self explanatory given the discussions of the Common Element Model and the Attribute Model. The entities in these two models are inherited by the EDIM, and additional entities are defined for the entities outcome_infers and related_inference_group. The outcome_infers entity is central to define the inferences that can be drawn from test outcomes. As described above, the DIM only permits two test outcomes–pass and fail. Further, the DIM differentiates between symmetric inferences, asymmetric inferences, and cross-linked inferences. The EDIM implicitly allows all of these through a general inference structure based on disjunctive normal form.

Disjunctive normal form (DNF) is a way of representing any boolean expression and consists of expressions of the form, $(p_1 \wedge p_2 \wedge \cdots) \vee (q_1 \wedge q_2 \wedge \cdots) \vee \cdots$. The Express entities for encoding the DNF inferences are somewhat difficult to follow, but they can be reduced to the representations in the DIM in a straightforward way. The outcome_infers and dependent entities adapted for the DIM are as follows:

```
ENTITY outcome_infers;
    for test:               test;
    for outcome:            select (pass, fail);
    infer:                  set [1:?] of product_term;
END_ENTITY;

ENTITY product_term;
    inference:              set [0:?] of term;
    negative_inference:     set [0:?] of term;
END_ENTITY;

TYPE  term  =  select  (test_inference,  diagnostic_inference,
    related_inference_group_inference);
END_TYPE;

ENTITY test_inference;
    for_test:               test;
    has_outcome:            select (pass, fail);
END_ENTITY;

ENTITY diagnostic_inference;
    for_diagnosis:          diagnosis;
    assert;                 select (pass, fail);
END_ENTITY;

ENTITY related_inference_group_inference;
    for_inference_group:    related_inference_group;
END_ENTITY;
```

The related_inference_group entity is used to provide a means of arbitrarily combining tests and diagnoses. The test and diagnosis entities are set up as part-whole hierarchies and do not permit groups of tests or diagnoses to overlap. In the case where we might want to arbitrarily group elements independent of the structure of the model, we can use this entity. This entity is not usually considered to be a normal part of the DIM, but its addition is straightforward. The related_inference_group entity is defined as follows:

```
ENTITY related_inference_group;
    name:                   string;
    description:            optional string;
    members:                set [1:?] of inference_element;
END_ENTITY;
```

```
TYPE inference_element = select (diagnostic_inference, test_inference);
END_TYPE;
```

VI. THE FUTURE OF AI-ESTATE

The AI-ESTATE subcommittee is currently focusing on two different models for the current release of the P1232.1 standard–the fault tree model and the EDIM. It is the intent of this committee to expand the scope of the standard to several other models of diagnostic knowledge including rule bases, constraint networks, neural networks, frames, etc. We envision the standard consisting of a base document outlining the objectives of the standard and a set of normative annexes providing the specifications for the individual models. The common element model and attribute model will most likely appear in the main body with the fault tree model and EDIM comprising the first two normative annexes. This architecture will permit additional forms of knowledge representation to be added to the standard without the need to reballot the standard with each addition–only the new annexes will need to be balloted.

In addition to its efforts in developing standard interchange formats for diagnostic knowledge, AI-ESTATE is also developing a set of service specifications for use by a test system to interact with a diagnostic reasoner. This standard (designated P1232.2) will permit AI-ESTATE conformant reasoners to be interchanged among testers claiming to use 1232.2. Currently, only initial proposals have been developed for this standard, so it is expected that the service specification will receive more attention in the near future.

VII. CONCLUSION

In this paper, we have outlined the efforts of the AI-ESTATE subcommittee of SCC20 to standardize interchange formats for diagnostic models in intelligent test systems. We also provided a theoretical basis for the primary model in the current release of the standard–the enhanced diagnostic inference model (EDIM). While the EDIM appears to have strong ties to the dependency model of the 1980s, it is our hope that people attempting to use the model will recognize the intended philosophy for applying the model in diagnosis and including the knowledge in a standard test environment. It is also the hope of the AI-ESTATE subcommittee and its working groups that people interested in the standard become directly involved in its development; we welcome all interested parties to attend the meetings of the committee.

ACKNOWLEDGMENT

Many thanks to the people who have been involved in writing the current draft of P1232.1, including William R. Simpson, Les Orlidge, Sharon Goodall, Bernard Dugas, and Jack Taylor. We also thank the U.S. Navy for providing valuable information on their efforts on the DIM handbook: Mike Lynch and Tim Bearse. Finally, we thank Harry Dill for providing much of the information that is currently being incorporated in the DIM handbook and for providing tremendous insight into the issues involved in developing accurate and useful diagnostic models.

REFERENCES

[1] IEEE Std 1226-1993, *IEEE Trial Use Standard for a Broad-Based Environment for Test, Overview and Architecture*, 1993.

[2] IEEE Std P1232, *IEEE Trial Use Standard for Artificial Intelligence and Expert System Tie to Automatic Test Equipment, Overview and Architecture*, Draft 5.0, May 1994.

[3] Sheppard, J. W., and G. C. Hadfield, "The Object Oriented Design of Intelligent Test Systems," *AUTOTESTCON 93 Conference Record*, San Antonio, TX, 1993.

[4] Simpson, W. R., and J. W. Sheppard, "Fault Isolation in an Integrated Diagnostic Environment," *IEEE Design and Test of Computers*, 10(1):52-66, March 1993.

[5] DePaul, R. Jr., "Logic Modeling as a Tool for Testability," *AUTOTESTCON '85 Symposium Proceedings*, Uniondale, New York, September 1985.

[6] Sheppard, J. W., and W. R. Simpson, "A Mathematical Model for Integrated Diagnostics," *IEEE Design and Test of Computers*, 8(4):25-38, December 1991.

[7] IEEE Std 716-1989, *IEEE Standard C/ATLAS Test Language*, 1989.

[8] IEEE Std P1232.1, *IEEE Trial Use Standard for Artificial Intelligence and Expert System Tie to Automatic Test Equipment, Data and Knowledge Specification*, Draft 3.2, April 1993.

[9] Simpson, W. R., and H. S. Balaban, "The ARINC System Testability and Maintenance Program (STAMP), *Proceedings of AUTOTESTCON '82*, Dayton, Ohio, 1982.

[10] Peng, Y., and J. A. Reggia, *Abductive Inference Models for Diagnostic Problem-Solving*, New York: Springer-Verlag, 1990.

[11] Pearl, P., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, San Mateo, California: Morgan-Kaufmann Publishers, 1988.

[12] Simpson, W. R., and J. W. Sheppard, *System Test and Diagnosis*, Kluwer Academic Publishers, in press.