IEEE Std 1522-2004

IEEE Standards

1522™

IEEE Trial-Use Standard for Testability and Diagnosability Characteristics and Metrics

IEEE Standards Coordinating Committee 20

Sponsored by the
IEEE Standards Coordinating Committee 20 on
Test and Diagnosis for Electronic Systems

IEEE

3 Park Avenue, New York, NY 10016-5997, USA

23 March 2005

Print: SH95278
PDF: SS95278

IEEE Trial-Use Standard for Testability and Diagnosability Characteristics and Metrics

Sponsored by

IEEE Standards Coordinating Committee 20 on Test and Diagnosis for Electronic Systems

Approved 23 September 2004

IEEE-SA Standards Board

Abstract: This standard was developed to provide standard, unambiguous definitions of testability and diagnosability metrics and characteristics. It builds on fundamental definitions derived from elements in formal information models related to test and diagnosis defined in IEEE Std 1232™-2002.
Keywords: aggregation fundamentals, cost-related fundamentals, diagnosability, diagnostic inference model, enhanced diagnostic inference model, fault tree model, metrics, testability

---

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Print: ISBN 0-7381-4492-4 SH95278
PDF: ISBN 0-7381-4493-2 SS95278

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

Use of an IEEE Standard is wholly voluntary. The IEEE disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other IEEE Standard document.

The IEEE does not warrant or represent the accuracy or content of the material contained herein, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained herein is free from patent infringement. IEEE Standards documents are supplied "**AS IS**."

The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

In publishing and making this document available, the IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other IEEE Standards document, should rely upon the advice of a competent professional in determining the exercise of reasonable care in any given circumstances.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position, explanation, or interpretation of the IEEE.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board

445 Hoes Lane

Piscataway, NJ 08855-1331USA

NOTE—Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents for which a license may be required by an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Introduction

This introduction is not part of IEEE Std 1522-2004, IEEE Trial-Use Standard for Testability and Diagnosability Characteristics and Metrics.

Publication of this trial-use standard for comment and criticism has been approved by the Institute of Electrical and Electronics Engineers. Trial-use standards are effective for 24 months from the date of publication. Comments for revision will be accepted for 18 months after publication. Suggestions for revision should be directed to the Secretary, IEEE-SA Standards Board, 445 Hoes Lane, Piscataway, NJ 08855-1331, and should be received no later than 23 March 2006. It is expected that following the 24-month period, this trial-use standard, revised as necessary, shall be submitted to the IEEE-SA Standards Board for approval as a full-use standard.

As systems became more complex, costly, and difficult to diagnose and repair, initiatives were started to address these problems. The objective of one initiative, testability, was to make systems easier to test. To be useful, this commercial standard must provide specific, unambiguous definitions of criteria for assessing system testability.

Recent initiatives by the Institute of Electrical and Electronics Engineers (IEEE), Inc. on standardizing test architectures have provided an opportunity to standardize testability metrics. The IEEE 1232™ Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE) series of standards provide the foundation for precise and unambiguous testability and diagnosability metrics.

It is the objective of this standard to provide notionally correct and inherently useful but mathematically precise definitions of testability measures that may be used to either measure the testability characteristics of a system or to predict the testability of a system. Notionally correct means that the measures are not in conflict with intuitive and historical representations.

Notice to users

Errata

Errata, if any, for this and all other standards can be accessed at the following URL: http://standards.ieee.org/reading/ieee/updates/errata/index.html. Users are encouraged to check this URL for errata periodically.

Interpretations

Current interpretations can be accessed at the following URL: http://standards.ieee.org/reading/ieee/interp/index.html.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents or patent applications for which a license may be required to implement an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Participants

The following is a list of participants in the Diagnostic & Maintenance Control (D&MC) subcommittee.

Mark A. Kaufman, *Co-chair*
Tim Wilmering, *Co-Chair*

Sherif Abdelwahed
Guy Adam
Greg P. Bowman
Eric M. Bukata
Antonius Bunsen
Stephen J. Cmiel
Bernard Dathy
Bernd Dinklage
David B. Droste
William B. Frank
George E. Geathers
Sudipto Ghoshal
Gerald M. Goldemund
Chris C. Gorringe
Eric S. Gould
Michael E. Keller
Leslie A. Orlidge
Mukund U. Modi
N. Ramachandran
Mike L. Seavey
John W. Sheppard
Li Pi Su
William J. Taylor
Joerg H. Urban
J. Richard Weger

Other persons who contributed to this trial-use standard are as follows:

Antony Bartolini
Amanda Jane Giarla

The following members of the individual balloting committee voted on this trial-use standard. Balloters may have voted for approval, disapproval, or abstention.

Martin Baur
Tommy Cooper
Em Delahostria
Bernd Dinklage
William Frank
Eric Gould
Randall Groves
Ashley Hulme
Mark Kaufman
Adam Ley
Gregory Luri
Gary Michel
Mukund Modi
Charles Ngethe
Paul Pillitteri
Vikram Punj
N. Ramachandran
Klaus Rapf
James Ruggieri
John Sheppard
James Smith
Timothy Wilmering

When the IEEE-SA Standards Board approved this trial-use standard on 23 September 2004, it had the following membership:

Don Wright, *Chair*
Steve M. Mills, *Vice Chair*
Judith Gorman, *Secretary*

Chuck Adams
H. Stephen Berger
Mark D. Bowman
Joseph A. Bruder
Bob Davis
Roberto de Marca Boisson
Julian Forster*
Arnold M. Greenspan
Mark S. Halpin
Raymond Hapeman
Richard J. Holleman
Richard H. Hulett
Lowell G. Johnson
Joseph L. Koepfinger*
Hermann Koch
Thomas J. McGean
Daleep C. Mohla
Paul Nikolich
T. W. Olsen
Ronald C. Petersen
Gary S. Robinson
Frank Stone
Malcolm V. Thaden
Doug Topping
Joe D. Watson

*Member Emeritus

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

Satish K. Aggarwal, *NRC Representative*
Richard DeBlasio, *DOE Representative*
Alan Cookson, *NIST Representative*

Don Messina
IEEE Standards Project Editor

Contents

IEEE Trial-Use Standard for Testability and Diagnosability Characteristics and Metrics

1. Overview

This standard was developed by the Diagnostic and Maintenance Control (D&MC) subcommittee of the IEEE Standards Coordinating Committee 20 (SCC20) on Test and Diagnosis for Electronic Systems to provide standard, unambiguous definitions of testability and diagnosability metrics and characteristics. This standard builds on fundamental definitions derived from elements in formal information models related to test and diagnosis defined in IEEE Std 1232™-2002.[1]

The goals of this standard are summarized as follows:

— Provide definitions of testability and diagnosability characteristics and metrics that are independent of specific test and diagnosis technologies.

— Provide definitions of testability and diagnosability characteristics and metrics that are independent of specific "system under test" technologies.

— Provide unambiguous definitions of testability and diagnosability metrics used by procurement and support organizations.

Over a century ago, Lord Kelvin commented: "When you can measure what you are speaking about and express it in numbers, you know something about it; and when you cannot measure it, you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind. It may be the beginning of knowledge, but you are scarcely in your thought advanced to the stage of a science." To add to Lord Kelvin's insight, when different tools measure something differently, and the basis for the measurement is not known or understood, it is neither knowledge nor science.

It is the objective of this standard to provide precise formal definitions of metrics that may be used to predict the testability of a system. Predictive metrics may be used iteratively during various phases of design and diagnostic development to map the progress toward a project's testability goals. On the other hand, empirical metrics are derived from actual maintenance data to determine the testability achieved during the factory and field testing of a system.

[1]Information for references can be found in Clause 2.

It is the intent of this standard that predictive testability and diagnosability metrics be defined based on IEEE Std 1232-2002. Where entities are required that have not been defined in these information models, this standard will provide its own information model in the form of an AI-ESTATE extension to satisfy the deficiency. It is not the intent of this standard to impose any implementation-specific requirements for actually computing the metrics; however, metrics not computed using the identified standard information models must be demonstrated to be mathematically equivalent to the definitions provided. Furthermore, metrics defined in this standard are based on the assumption that no more than a single independent fault exists at the time of diagnostics. Multiple fault metrics are outside the scope of this standard because of a lack of industry consensus as to how those metrics should be calculated.

1.1 Scope

This standard defines technology-independent testability and diagnosability characteristics and metrics, particularly those based on relevant standard information models, including standard AI-ESTATE information models.

1.2 Purpose

This standard will provide consistent, unambiguous definitions of testability and diagnosability characteristics and metrics. As a result, a common basis for system testability and diagnosability assessment will be provided.

1.3 Conventions used in this standard

The subclauses present specification formats using the EXPRESS language (ISO 10303-11:1994). All specifications in the EXPRESS language are given in the Courier type font, which includes references to entity and attribute names in the supporting text.

This standard uses the vocabulary and definitions of relevant IEEE standards. In case of conflict of definitions, except for those portions quoted from standards, the following precedence shall be observed: (1) Clause 3; (2) SCC20 documentation and standards; and 3) *The Authoritative Dictionary of IEEE Standards Terms*, seventh edition [B4]. For clarity, portions of IEEE Std 1232-2002 are repeated in this standard. In the event of revision to AI-ESTATE, the current, approved version of IEEE Std 1232-2002 takes precedence.

2. References

This standard shall be used in conjunction with the following publications.

IEEE Std 1232-2002, IEEE Standard for Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE).[2]

ISO 10303-11:1994, Industrial Automation Systems and Integration—Product Data Representation and Exchange—Part 11: Description Methods: The EXPRESS Language Reference Manual.[3]

[2]IEEE publications are available from the Institute of Electrical and Electronics Engineers, Inc., 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org/).

[3]ISO publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/ Suisse (http://www.iso.ch/). ISO publications are also available in the United States from the Sales Department, American National Standards Institute, 25 West 43rd Street, 4th Floor, New York, NY 10036, USA (http://www.ansi.org/).

3. Definitions

For the purposes of this standard, the following terms and definitions apply. *The Authoritative Dictionary of IEEE Standards and Terms*, Seventh edition [B4] should be referenced for terms not defined in this clause.[4]

3.1 characteristic: With respect to testability and diagnosability, this property or indicator of a system describes its behavior under test. It may or may not be measurable during actual system operation or maintenance.

3.2 diagnostic strategy: An approach taken to combine factors, including constraints, goals, and other considerations to be applied to localizing faults in a system.

3.3 diagnosability: The degree to which faults within a system can be *confidently* and *efficiently* identified. Confidently means frequently and unambiguously detecting and isolating faults when they occur. Efficiently means optimizing the resources required to achieve fault isolation.

3.4 fault detection: The process of identifying and reporting the presence of one or more faults within a system.

3.5 metric: With respect to testability and diagnosability, this characteristic can be evaluated by analysis or measured during system operation and maintenance.

3.6 testability: A system design characteristic that allows its operational status to be determined and the isolation of faults to be performed efficiently.

4. Context

As systems became more complex, costly, and difficult to diagnose and repair, initiatives were started to address these problems. Early on, these initiatives focused on having enough test points in the right places. As systems evolved, it was recognized that the system design had to include characteristics to make the system easier to test. This recognition was the start of considering testability as a design characteristic. As defined in MIL-STD-2165, testability is "a *design characteristic* which allows the status (operable, inoperable, or degraded) of an item to be determined and the isolation of faults within the item to be performed in a timely manner" ([B1], emphasis added). The purpose of MIL-STD-2165 was to provide uniform procedures and methods to control planning, implementation, and verification of testability during the system acquisition process. This standard, although deficient in some areas, provided useful guidance to government suppliers. Furthermore, lacking any equivalent industry standard, many commercial system developers have used it to guide their activities, although it was not imposed as a requirement.

Today, the discipline of testability is made up of two related practices. The first of these, design for test (DFT), facilitates the testing process by encouraging good design practices. The second testability practice, the one that is addressed within this standard, evaluates *diagnosability*. Thus far, formal, unambiguous definitions of testability and diagnosability metrics and characteristics have been unavailable. This standard addresses this issue by building on the fundamental definitions in the information models of AI-ESTATE. Specifically, this standard defines metrics related to fault detection and isolation.

Traditionally, fault detection metrics are represented as a ratio of detected or detectable faults to the full set of possible faults under consideration (often called a fault universe). Frequently, this ratio is weighted using the failure rates of the faults within the fault universe so that it expresses the likelihood of a fault being detected.

[4]The numbers in brackets correspond to those of the bibliography in Annex A.

Traditionally, fault isolation metrics are based on a ratio of the number of faults in ambiguity groups of a specified size (or less) to the total number of faults that can be isolated for a given system. As with detection, isolation calculations are frequently weighted using failure rates so that they express the likelihood of isolating a fault within an ambiguity group of a given size.

In addition to fault detection and fault isolation, testability has been concerned with, among other things, attempting to predict and measure the incidence of false indication (i.e., frequency of false alarm and nondetection). False indication issues are complex, and no consensus exists in the test industry for predicting and measuring the incidence of false indication. Given this lack of consensus, this standard will defer specification of metrics for false indication until such time as consensus is reached.

4.1 Predictive and empirical metrics

This standard distinguishes between two classes of metrics—predictive and empirical. Predictive metrics include all testability and diagnosability metrics that can be derived from a static AI-ESTATE-compliant model (i.e., the fault tree model [FTM], diagnostic inference model [DIM], or enhanced diagnostic inference model [EDIM]) and thus provide a quantitative evaluation of how well the system can be diagnosed using that diagnostic model. These metrics will be calculated differently depending on whether the diagnostic model is developed in the form of a fault tree (FTM) or test-to-diagnosis inference relationships (DIM or EDIM). Empirical metrics, as previously discussed, are derived from collected maintenance data and are beyond the scope of this standard.

In defining the metrics for this standard, the following approach has been taken. Clause 5 provides several fundamental measures to be used in defining testability and diagnosability metrics. First, several fundamental functions are defined in 5.2, which aggregate basic elements from the AI-ESTATE models. Next, functions are defined in 5.3 to estimate cost of tests or probability of faults based on the cost model within AI-ESTATE.[5] Third, several functions are defined in 5.4 to assist in traversing the diagnostic models within AI-ESTATE. Although all of the fundamental measures under Clause 5 are defined using the AI ESTATE information models, often multiple correct means compute the individual fundamentals. As it is the purpose of this standard to provide sematically rigorous definitions, not to dictate the method(s) of calculation, detailed algorithms are not provided for these fundamental measures. Finally, Clause 6 uses these fundamental measures from Clause 5 to define the standard metrics for testability and diagnosability.

4.2 Relationship to AI-ESTATE standard

IEEE Std 1232-2002 (AI-ESTATE) is an information exchange standard for test and diagnosis. The standard defines the architecture of an AI-ESTATE-conformant system, the information models for knowledge and data exchange, and a set of standard software services to be provided by a diagnostic reasoner in an open-architecture test environment. Because it provides formal definitions (via information modeling) of the information required for test and diagnosis—the same information required for determining the testability and diagnosability of a system—the AI-ESTATE standard is used as the basis for the metrics defined within this standard.

A description of how to use the fundamentals to develop metrics is contained in Clause 6.

4.3 Conformance

This subclause defines the requirements for conformance to the standard testability and diagnosability characteristics and metrics defined in this standard. Conformance demands that all metrics be computed as specified in Clause 6.

[5]Failure rate was accounted for by extending the AI-ESTATE models with the EXTEND schema in 5.3.1.

4.4 Extensibility

Although extensions are nonconformant, an extensibility mechanism is provided to facilitate future enhancement of this standard. Extensions should be submitted to the D&MC Subcommittee to be considered for inclusion in future versions of the standard. Proposed extensions to the metrics defined in this standard are ranked in order of preference as follows:

a) Metrics derived strictly from the fundamental measures of Clause 5

b) Metrics derived from functions and measures based strictly on the AI-ESTATE models

c) Metrics derived from functions and measures based on the AI-ESTATE models in combination with AI-ESTATE-conformant extensions (i.e., EXTEND schemata)

Extensions cannot be used to redefine any metrics defined in this standard.

5. Fundamental measures

The fundamental measures defined in this clause are categorized as either aggregation measures, cost-related measures, or measures to support detection and isolation metrics. The specific measures defined in this clause are listed in Table 1.

5.1 Overview and assumptions

The approach being taken to define standard, unambiguous testability and diagnosability metrics is based on the assumption that standard information models provide formal semantics for information of interest to the testability and diagnosability domain. In this section, several specific, fundamental measures will be defined using terminology from EXPRESS models defined in AI-ESTATE. The approach to specifying measures will follow the conventions for defining services in AI-ESTATE (see 4.2).

The fundamental measures in this clause are based on the following propositions:

— The primary information models used to define these measures are drawn from the AI-ESTATE standard. The specific models used include the AI-ESTATE common element model, fault tree model, diagnostic inference model, and enhanced diagnostic inference model.

— All cost-related metrics shall be computed with respect to a single test or repair within the AI-ESTATE common element model as qualified by specifically identified constraints and types.

— To manage the complexity of computing isolation-related metrics and to remove dependence on diagnostic and maintenance strategies, all isolation metrics shall be computed under the assumption that no more than a single, independent fault exists at the time of diagnostics.

— When defining the measures for detection and isolation, it is clear that the type of underlying diagnostic model (i.e., fault tree model, diagnostic inference model, or enhanced diagnostic inference model) will establish detectability and isolatability differently. Consequently, these measures will be defined with an identification of which of the model types provided in AI-ESTATE is being used.

— Many measures are defined relative to a “level” defined within the AI-ESTATE model. AI-ESTATE defines the level entity to be “the level of indenture for a particular hierarchical element. This defines a common level for purposes of constraining diagnosis or analysis for testability and diagnosability assessment.”

Table 1—Defined fundamental measures

Aggregation functions
action_set
diagnosis_set
failure_mode_set
failure_set
fault_set
function_set
repair_item_set
repair_set
resource_set
Cost functions
probability_of_diagnosis
probability_of_repair_item_set
repair_context_non_time_cost
repair_context_time_cost
repair_level_non_time_cost
repair_level_time_cost
repair_non_time_cost
repair_time_cost
test_context_non_time_cost
test_context_time_cost
test_level_non_time_cost
test_level_time_cost
test_non_time_cost
test_time_cost
total_repair_non_time_cost
total_repair_time_cost
total_test_non_time_cost
total_test_time_cost
Detection and isolation functions
get_leaves
get_signature
get_ambig_group
get_diag_ambig_group
get_size_n_ambig_group
get_size_n_diag_ambig_group
convert_repair_to_diag_ambiguity
convert_diag_to_repair_ambiguity
repair_item_ambiguity
diagnosis_ambiguity
single_level_detection
implicated_diagnosis_set
detectable_failure_set
detectable_fault_set
isolatable_repair_item_set
isolatable_diagnosis_set
isolatable_failures_set
isolatable_faults_set
fault_ambig_set
failure_ambig_set

5.2 Aggregation fundamentals

This subclause provides the definition of each aggregation fundamental to be used in more complex metric definitions. All fundamental measures are based on the AI-ESTATE common element model and need not be tied to specific diagnostic models (i.e., FTM, DIM, or EDIM).

5.2.1 action_set

Within AI-ESTATE, an action is defined by the information model to be "A specific action taken to either test or repair an item. Actions are the primary entities against which costs are associated. Ultimately, the diagnosis and repair process is concerned with optimizing the sequence of actions required to return a unit to service." The function `action_set` returns the set of all actions within a diagnostic model at a particular level.

```
FUNCTION action_set
   (mdl : diagnostic_model;
 lvl : level) : SET OF action;
END_FUNCTION;
```

5.2.2 diagnosis_set

Within AI-ESTATE, a diagnosis is defined by the information model to be "a lattice of diagnostic conclusions. Typically, a diagnosis consists of failures or fault (*sic*.). The diagnoses can be interconnected in a lattice to classify constituent units, e.g., by function. This construct provides a generalized grouping mechanism for the constituents of the unit under test and their failure modes." The function `diagnosis_set` returns the set of all diagnoses within a diagnostic model at a particular level.

```
FUNCTION diagnosis_set
   (mdl : diagnostic_model;
    lvl : level) : SET OF diagnosis;
END_FUNCTION;
```

5.2.3 failure_mode_set

Within AI-ESTATE, a failure mode is defined by the information model to be "A specific mode or means by which a unit or system can fail. The specific failure mode is associated, first with the diagnosis, then (by way of the appropriate subtype) to the item that has failed." The function `failure_mode_set` returns the set of all failure modes within a diagnostic model at a particular level. As failure modes are associated with diagnoses within the model, level is determined by examining appropriate diagnoses.

```
FUNCTION failure_mode_set
   (mdl : diagnostic_model;
    lvl : level) : SET OF failure_mode;
END_FUNCTION;
```

5.2.4 failure_set

Within AI-ESTATE, a failure is defined by the information model to be “A manifestation of an anomaly within a system. When considering a functional model, it is the failure of the system under test to perform some intended function.” The function `failure_set` returns the set of all failures within a diagnostic model at a particular level.

```
FUNCTION failure_set
   (mdl : diagnostic_model;
    lvl : level) : SET OF failure;
END_FUNCTION;
```

5.2.5 fault_set

Within AI-ESTATE, a fault is defined by the information model to be “A physical cause of anomalous behavior within a system. A fault typically corresponds to some physical breakdown in the system under test.” The function `fault_set` returns the set of all faults within a diagnostic model at a particular level.

```
FUNCTION fault_set
   (mdl : diagnostic_model;
    lvl : level) : SET OF fault;
END_FUNCTION;
```

5.2.6 function_set

Within AI-ESTATE, a function is defined by the information model to be “any ‘functional behavior’ represented within a system. The function entity is intended to provide a placeholder for functional information in the event the diagnostic or testability model is function-oriented.” The function `function_set` returns the set of all functions within a diagnostic model at a particular level.

```
FUNCTION function_set
   (mdl : diagnostic_model;
    lvl : level) : SET OF func;
END_FUNCTION;
```

5.2.7 repair_item_set

Within AI-ESTATE, a repair item is defined by the information model to be “some part of the system to be adjusted, calibrated, repaired, replaced, etc. In short, a diagnosis shall lead the maintainer to the object of the maintenance action.” The function `repair_item_set` returns the set of all repair items within a diagnostic model at a particular level.

```
FUNCTION repair_item_set
   (mdl : diagnostic_model;
    lvl : level) : SET OF repair_item;
END_FUNCTION;
```

5.2.8 repair_set

Within AI-ESTATE, a repair is defined by the information model to be the process undertaken to "return the unit to service." The function `repair_set` returns the set of all repairs within a diagnostic model at a particular level.

```
FUNCTION repair_set
   (mdl : diagnostic_model;
    lvl : level) : SET OF repair;
END_FUNCTION;
```

5.2.9 resource_set

Within AI-ESTATE, a resource is defined within the information model to be something that "any given action in the maintenance process is likely to require … to perform that action." The function `resource_set` returns the set of all resources within a diagnostic model at a particular level.

```
FUNCTION resource_set
   (mdl : diagnostic_model;
    lvl : level) : SET OF resource;
END_FUNCTION;
```

5.3 Cost-related fundamentals

This subclause provides the definition of each cost-related fundamental to be used in more complex metric definitions. All fundamental measures are based on the AI-ESTATE common element model and need not be tied to specific diagnostic models (i.e., FTM, DIM, and EDIM).

This standard assumes costs are measurable quantities of time or resources (e.g., money and/or expendables) to be managed in performing an action or sequence of actions. In defining the cost-related measures, this standard applies the categories of cost types as specified in AI-ESTATE, namely, time cost and non-time cost.

The fundamental measures in 5.3.3–5.3.18 are based on the following proposition:

— Cost is computed over a set of tests or repairs by using associated test action costs, repair action costs, and resource costs.

— All cost calculations use the same units of measure and are based on resource cost. No unit conversion is included in the definition.

Four versions of the measures (with the exception of probability_of_diagnosis and probability_of_repair_item_set) are specified—one associated with a particular level and a particular required context (as defined by the context entity within AI-ESTATE), one associated with a particular level but all contexts (i.e., no required context specified), one associated with a particular required context but all levels (i.e., no level specified), and one with no level and no required context specified. The context refers to the required set of conditions that is true when specified test, diagnosis, or repair is performed.

5.3.1 probability_of_diagnosis

Within AI-ESTATE, a model structure is provided to capture failure rate information for a particular diagnosis in a diagnostic model. This entity specifies whether the failure rate is defined based on “clock” time or “operating” time, where “clock” time includes the time when the system is not in operation and “operating” time does not. The time basis attribute also permits specification of a user-defined basis or specification that the basis is unknown. However, IEEE 1232-2002 (AI-ESTATE) lacks an attribute required for calculating the probability of diagnosis. To provide the required attribute, an extension to AI-ESTATE has been specified in Annex C.

The function `probability_of_diagnosis` returns the conditional probability of a diagnosis given a specified set of diagnoses. Typically, this probability is calculated as the failure rate of the specified diagnosis, divided by the sum of the failure rates of the diagnoses in the specified set. As an alternative, the underlying distribution (e.g., exponential or Weibull) would be used to derive a probability at a specified point in time for the members of the set, and the probability of diagnosis would be computed by normalizing over the specified set. In implementing this measure, the approach to calculating the measure must be specified. To determine probability of diagnosis, the following conditions must hold true:

— The probability of diagnosis is computed relative to a specified set of diagnoses.

— The diagnosis for which the probability is being calculated is a member of the specified diagnosis set.

— No diagnosis in the relative diagnosis set is a child of another diagnosis in the specified diagnosis set.

— The time basis is consistent for all members of the diagnosis set.

— The set of diagnoses does not mix faults and failures.

— The diagnoses in the specified set are probabilistically independent of one another.

The probability is then calculated using the extended diagnosis entity defined in Annex C.

```
FUNCTION probability_of_diagnosis
   (diag : EXTEND_diagnosis;
    diag_set : SET OF EXTEND_diagnosis;
    time_point : OPTIONAL cost_value;
    units : OPTIONAL time_unit) : REAL;
END_FUNCTION;
```

5.3.2 probability_of_repair_item_set

The measure `probability_of_repair_item_set` represents the probability that a set of repair items is diagnosed. Specifically, given a model, the probability of a repair item is defined as the sum of the “probabilities of diagnosis” for all diagnoses tied to that repair item. The probability of a set of repair items (such as might be found in a repair item ambiguity group) is the sum of the probabilities of the *set* of all diagnoses tied to the entire set of repair items. To determine probability of repair item, the following assumptions are made:

— No diagnosis in the repair item is a child of another diagnosis in the same repair item.

— The time basis is consistent for all diagnoses in the repair item.

— The set of diagnoses in the repair item does not mix faults and failures.

— Computing the probability of a diagnosis uses the probability_of_diagnosis function defined here.

— The diagnoses in the specified set are probabilistically independent of one another.

```
FUNCTION probability_of_repair_item_set
   (rpr : SET [1:?] OF repair_item;
    diag_set : SET OF EXTEND_diagnosis;
    time_point : OPTIONAL cost_value;
    units : OPTIONAL time_unit) : REAL;
END_FUNCTION;
```

5.3.3 repair_context_non_time_cost

The cost-related measure `repair_context_non_time_cost` adds the non-time cost of all actions (with the costs of the actions' required resources included) associated with performing the indicated repair, limited to the resources and actions available in the required context.

NOTE—This function computes repair cost given the specified context.

```
FUNCTION repair_context_non_time_cost
   (rpr : SET OF repair;
    cntxt : required_context;
 of_type : cost_type) : REAL;
END_FUNCTION;
```

5.3.4 repair_context_time_cost

The cost-related measure `repair_context_time_cost` adds the time cost of all actions (with the costs of the actions' required resources included) associated with performing the indicated repair, limited to the resources and actions available in the required context.

NOTE—This function computes repair cost given the specified context.

```
FUNCTION repair_context_time_cost
   (rpr : SET OF repair;
    cntxt : required_context;
 of_type : cost_type) : REAL;
END_FUNCTION;
```

5.3.5 repair_level_non_time_cost

The cost-related measure `repair_level_non_time_cost` adds the non-time cost of all actions (with the costs of the actions' required resources included) associated with performing the indicated repair, limited to the resources and actions available at a particular level of indenture.

NOTE—This function computes repair cost given the specified level.

```
FUNCTION repair_level_non_time_cost
   (rpr : SET OF repair;
    lvl : level;
 of_type : cost_type) : REAL;
END_FUNCTION;
```

5.3.6 repair_level_time_cost

The cost-related measure `repair_level_time_cost` adds the time cost of all actions (with the costs of the actions' required resources included) associated with performing the indicated repair, limited to the resources and actions available at a particular level of indenture.

NOTE—This function computes repair cost given the specified level.

```
FUNCTION repair_level_time_cost
   (rpr : SET OF repair;
    lvl : level;
 of_type : cost_type) : REAL;
END_FUNCTION;
```

5.3.7 repair_non_time_cost

The cost-related measure `repair_non_time_cost` adds the non-time cost of all actions (with the costs of the actions' required resources included) associated with performing the indicated repair, limited to the resources and actions available in the required context at a particular level of indenture.

NOTE—This function computes repair cost given both the specified context and the specified level.

```
FUNCTION repair_non_time_cost
   (rpr : SET OF repair;
    cntxt : required_context;
 lvl : level;
 of_type : cost_type) : REAL;
END_FUNCTION;
```

5.3.8 repair_time_cost

The cost-related measure `repair_time_cost` adds the time cost of all actions (with the costs of the actions' required resources included) associated with performing the indicated repair, limited to the resources and actions available in the required context at a particular level of indenture.

NOTE—This function computes repair cost given both the specified context and the specified level.

```
FUNCTION repair_time_cost
   (rpr : SET OF repair;
    cntxt : required_context;
 lvl : level;
 of_type : cost_type) : REAL;
END_FUNCTION;
```

5.3.9 test_context_non_time_cost

The cost-related measure `test_context_non_time_cost` adds the non-time cost of all actions (with the costs of the actions' required resources included) associated with performing the indicated test, limited to the resources and actions available in the required context.

NOTE—This function computes test cost given the specified context.

```
FUNCTION test_context_non_time_cost
   (tst : SET OF test;
    cntxt : required_context;
 of_type : cost_type) : REAL;
END_FUNCTION;
```

5.3.10 test_context_time_cost

The cost-related measure `test_context_time_cost` adds the time cost of all actions (with the costs of the actions' required resources included) associated with performing the indicated test, limited to the resources and actions available in the required context.

NOTE—This function computes test cost given the specified context.

```
FUNCTION test_context_time_cost
   (tst : SET OF test;
    cntxt : required_context;
 of_type : cost_type) : REAL;
END_FUNCTION;
```

5.3.11 test_level_non_time_cost

The cost-related measure `test_level_non_time_cost` adds the non-time cost of all actions (with the costs of the actions' required resources included) associated with performing the indicated test, limited to the resources and actions available at a particular level of indenture.

NOTE—This function computes test cost given the specified level.

```
FUNCTION test_level_non_time_cost
   (tst : SET OF test;
    lvl : level;
 of_type : cost_type) : REAL;
END_FUNCTION;
```

5.3.12 test_level_time_cost

The cost-related measure `test_level_time_cost` adds the time cost of all actions (with the costs of the actions' required resources included) associated with performing the indicated test, limited to the resources and actions available at a particular level of indenture.

NOTE—This function computes test cost given the specified level.

```
FUNCTION test_level_time_cost
   (tst : SET OF test;
    lvl : level;
 of_type : cost_type) : REAL;
END_FUNCTION;
```

5.3.13 test_non_time_cost

The cost-related measure `test_non_time_cost` adds the non-time cost of all actions (with the costs of the actions' required resources included) associated with performing the indicated test, limited to the resources and actions available in the required context at a particular level of indenture.

NOTE—This function computes test cost given both the specified context and the specified level.

```
FUNCTION test_non_time_cost
   (tst : SET OF test;
    cntxt : required_context;
 lvl : level;
 of_type : cost_type) : REAL;
END_FUNCTION;
```

5.3.14 test_time_cost

The cost-related measure `test_time_cost` adds the time cost of all actions (with the costs of the actions' required resources included) associated with performing the indicated test, limited to the resources and actions available in the required context at a particular level of indenture.

NOTE—This function computes test cost given both the specified context and the specified level.

```
FUNCTION test_time_cost
   (tst : SET OF test;
    cntxt : required_context;
 lvl : level;
 of_type : cost_type) : REAL;
END_FUNCTION;
```

5.3.15 total_repair_non_time_cost

The cost-related measure `total_repair_non_time_cost` adds the non-time cost of all actions (with the costs of the actions' required resources included) associated with performing the indicated repair, with no limits to the resources and actions available based on required level or context.

```
FUNCTION total_repair_non_time_cost
   (rpr : SET OF repair;
    of_type : cost_type) : REAL;
END_FUNCTION;
```

5.3.16 total_repair_time_cost

The cost-related measure `total_repair_time_cost` adds the time cost of all actions (with the costs of the actions' required resources included) associated with performing the indicated repair, with no limits to the resources and actions available based on required level or context.

```
FUNCTION total_repair_time_cost
   (rpr : SET OF repair;
    of_type : cost_type) : REAL;
END_FUNCTION;
```

5.3.17 total_test_non_time_cost

The cost-related measure `total_test_non_time_cost` adds the non-time cost of all actions (with the costs of the actions' required resources included) associated with performing the indicated test, with no limits to the resources and actions available based on required level or context.

```
FUNCTION total_test_non_time_cost
   (tst : SET OF test;
    of_type : cost_type) : REAL;
END_FUNCTION;
```

5.3.18 total_test_time_cost

The cost-related measure `total_test_time_cost` adds the time cost of all actions (with the costs of the actions' required resources included) associated with performing the indicated test, with no limits to the resources and actions available based on required level or context.

```
FUNCTION total_test_time_cost
   (tst : SET OF test;
    of_type : cost_type) : REAL;
END_FUNCTION;
```

5.4 Detection and isolation

This subclause defines each detection and isolation fundamental to be used in more complex metric definitions. Note that because each respective type of diagnostic models (i.e., FTM, DIM, or EDIM) is a subtype of the diagnostic_model entity in the AI-ESTATE common element model, it is sufficient to test for type of model in the functions for detectability and isolatability. Furthermore, note that as specified in AI-ESTATE, it is assumed that the diagnostic models used to specify these measures include an instance of the diagnosis entity corresponding to No Fault. As indicated, this entity shall have the unique name, "`no_fault.`"

Fundamental to the ability to define metrics based on detection and isolation within a diagnostic model is the ability to identify all diagnoses that are detectable or isolatable by the set of tests included in the model. Also fundamental to the ability to define these metrics in an unambiguous way is the requirement that all metrics are to be computed based on a predefined universe of diagnoses and a predefined set of tests. Without such a requirement, there is no way to ensure agreement and repeatability in determining the detection or isolation measures.

Note that actual measures of isolation depend on the algorithms used to perform diagnosis (in addition to the model used). Therefore, the following measures, when implemented, shall be documented in a way that identifies the nature of their dependence on the underlying diagnostic algorithm and that such documentation is provided to users of the measures.

To support definition of fundamental measures, standard metrics, and extended metrics (see 4.3), several functions are defined in 5.4.1–5.4.20. To support these functions, three derived types are defined to represent ambiguity groups (for diagnoses or repair items) and test signatures.

```
TYPE diagnosis_ambiguity = SET OF diagnosis;
END_TYPE;

TYPE repair_ambiguity = SET OF repair_item;
END_TYPE;
```

```
TYPE signature = SET OF test_outcome;
END_TYPE;
```

5.4.1 get_leaves

The function `get_leaves` takes a given step from a fault tree model as specified by AI-ESTATE and returns a set of sets of diagnoses that reside at the leaves of the corresponding subtree. In other words, this function returns the set of diagnosis ambiguity groups within the subtree.

```
FUNCTION get_leaves
   (step : fault_tree_step) : SET OF diagnosis_ambiguity;
END_FUNCTION;
```

5.4.2 get_signature

The function `get_signature` takes a diagnosis as defined in 5.2.2 and returns the set of test outcomes that in combination identify the specified diagnosis. This set of test outcomes is referred to as the “test signature” for the diagnosis.

```
FUNCTION get_signature
   (mdl : diagnostic_model;
    diag : diagnosis) : signature;
END_FUNCTION;
```

5.4.3 get_ambig_group

The function `get_ambig_group` takes a specific repair item as defined in 5.2.7 and returns a set of repair item ambiguity groups containing repair items that are ambiguous with it, given a diagnostic model and level of indenture. Note that the returned set includes the originally specified repair item. The model specified must be a DIM, a FTM, or an EDIM.

```
FUNCTION get_ambig_group
   (mdl : diagnostic_model;
    rpr : repair_item;
    lvl : level) : SET OF repair_ambiguity;
END_FUNCTION;
```

5.4.4 get_diag_ambig_group

The function `get_diag_ambig_group` takes a specific diagnosis as defined in 5.2.2 and returns a set of diagnosis ambiguity groups containing diagnoses that are ambiguous with it, given a diagnostic model and level of indenture. Note that the returned set includes the originally specified diagnosis. The model specified must be a DIM, a FTM, or an EDIM.

```
FUNCTION get_diag_ambig_group
   (mdl : diagnostic_model;
    diag : diagnosis;
    lvl : level) : SET OF diagnosis_ambiguity;
END_FUNCTION;
```

5.4.5 get_size_n_ambig_group

The function `get_size_n_ambig_group` returns a set of repair item ambiguity groups, each of which contain exactly *n* repair items as defined in 5.2.7, given a diagnostic model and level of indenture. The model specified must be a DIM, a FTM, or an EDIM.

```
FUNCTION get_size_n_ambig_group
   (mdl : diagnostic_model;
    lvl : level;
    n : INTEGER) : SET OF repair_ambiguity;
END_FUNCTION;
```

5.4.6 get_size_n_diag_ambig_group

The function `get_size_n_diag_ambig_group` returns a set of diagnosis ambiguity groups, each of which contain exactly *n* diagnoses as defined in 5.2.2, given a diagnostic model and level of indenture. The model specified must be a DIM, a FTM, or an EDIM.

```
FUNCTION get_size_n_diag_ambig_group
   (mdl : diagnostic_model;
    lvl : level;
    n : INTEGER) : SET OF diagnosis_ambiguity;
END_FUNCTION;
```

5.4.7 convert_repair_to_diag_ambiguity

The function `convert_repair_to_diag_ambiguity` returns the set of diagnosis ambiguity groups, given a diagnostic model, level of indenture, and a set of repair item ambiguity groups, where diagnosis and repair item are defined in 5.2.2 and 5.2.7, respectively. The model specified must be a DIM, a FTM, or an EDIM.

```
FUNCTION convert_repair_to_diag_ambiguity
                     (mdl : diagnostic_model;
                            lvl : level;
    rag : SET OF repair_ambiguity) : SET OF diagnosis_ambiguity;
END_FUNCTION;
```

5.4.8 convert_diag_to_repair_ambiguity

The function `convert_diag_to_repair_ambiguity` returns the set of repair item ambiguity groups, given a diagnostic model, level of indenture, and a set of diagnosis ambiguity groups, where diagnosis and repair item are defined in 5.2.2 and 5.2.7, respectively. The model specified must be a DIM, a FTM, or an EDIM.

```
FUNCTION convert_diag_to_repair_ambiguity
                     (mdl : diagnostic_model;
                            lvl : level;
    dag : SET OF diagnosis_ambiguity) : SET OF repair_ambiguity;
END_FUNCTION;
```

5.4.9 repair_item_ambiguity

The predicate `repair_item_ambiguity` takes as input a diagnostic model, a target repair_item, a level of indenture, and a group size *n*, and it returns TRUE if the repair item exists in a repair item ambiguity group of exactly size *n*. It returns FALSE otherwise. The model specified must be a DIM, a FTM, or an EDIM.

```
FUNCTION repair_item_ambiguity
   (mdl : diagnostic_model;
    rpr : repair_item;
    lvl : level;
    n : INTEGER) : BOOLEAN;
END_FUNCTION;
```

5.4.10 diagnosis_ambiguity

The predicate `diagnosis_ambiguity` takes as input a diagnostic model, a target diagnosis (or subtype), a level of indenture, and a group size *n*, and it returns TRUE if the diagnosis exists in a diagnosis ambiguity group of exactly size *n*. It returns FALSE otherwise. The model specified must be a DIM, a FTM, or an EDIM.

```
FUNCTION diagnosis_ambiguity
   (mdl : diagnostic_model;
    diag : diagnosis;
    lvl : level;
    n : INTEGER) : BOOLEAN;
END_FUNCTION;
```

5.4.11 single_level_detection

The function `single_level_detection` takes as input a set of diagnoses (including faults and failures) and a level of indenture, and it returns a reduced set of diagnoses such that no diagnosis has a child diagnosis at the same level. In other words, this function finds all diagnoses at a specified level of indenture that have no parents in the diagnosis hierarchy at the same level of indenture.

```
FUNCTION single_level_detection
   (diag : SET OF diagnosis;
    lvl : level) : SET OF diagnosis;
END_FUNCTION;
```

5.4.12 implicated_diagnosis_set

The fundamental function `implicated_diagnosis_set` examines a diagnostic model and returns the set of implicated diagnoses within that model at a specified level of indenture, given the tests defined for that model. An implicated diagnosis, in this context, is a particular candidate diagnosis identified as a candidate explanation associated with an indicated test failure. The model specified must be a DIM, a FTM, or an EDIM.

```
FUNCTION implicated_diagnosis_set
   (mdl : diagnostic_model;
    lvl : level) : SET OF diagnosis;
END_FUNCTION;
```

5.4.13 detectable_failure_set

The fundamental function `detectable_failure_set` examines a diagnostic model and returns the set of detectable failures, which are a subtype of diagnosis, within that model at a specified level of indenture, given the tests defined for that model. By definition, a detectable failure is also an implicated diagnosis; thus, given the same set of tests, the detectable failure set will be a subset of the implicated diagnosis set. The model specified must be a DIM, a FTM, or an EDIM.

```
FUNCTION detectable_failure_set
   (mdl : diagnostic_model;
   lvl : level) : SET OF diagnosis;
END_FUNCTION;
```

5.4.14 detectable_fault_set

The fundamental function `detectable_fault_set` examines a diagnostic model and returns the set of detectable faults, which are a subtype of diagnosis, within that model at a specified level of indenture, given the tests defined for that model. By definition, a detectable fault is also an implicated diagnosis; thus, given the same set of tests, the detectable fault set will be a subset of the implicated diagnosis set. The model specified must be a DIM, a FTM, or an EDIM.

```
FUNCTION detectable_fault_set
   (mdl : diagnostic_model;
   lvl : level) : SET OF diagnosis;
END_FUNCTION;
```

5.4.15 isolatable_repair_item_set

The fundamental function `isolatable_repair_item_set` takes as input a diagnostic model, a level of indenture, and a group size *n*, and it returns the set of repair item ambiguity groups of exactly size *n*. An additional BOOLEAN argument is provided as input to indicate whether to include ambiguity with No Fault in the analysis. The model specified must be a DIM, a FTM, or an EDIM.

```
FUNCTION isolatable_repair_item_set
   (mdl : diagnostic_model;
    lvl : level;
    n : INTEGER;
    include_nf : BOOLEAN) : SET OF repair_ambiguity;
END_FUNCTION;
```

5.4.16 isolatable_diagnosis_set

The fundamental function `isolatable_diagnosis_set` takes as input a diagnostic model, a level of indenture, and a group size *n*, and it returns the set of diagnosis ambiguity groups of exactly size *n*. An additional BOOLEAN argument is provided as input to indicate whether to include ambiguity with No Fault in the analysis. The model specified must be a DIM, a FTM, or an EDIM.

```
FUNCTION isolatable_diagnoses_set
   (mdl : diagnostic_model;
    lvl : level;
    n : INTEGER;
   include_nf : BOOLEAN) : SET OF diagnosis_ambiguity;
END_FUNCTION;
```

5.4.17 isolatable_failures_set

The fundamental function `isolatable_failures_set` takes as input a diagnostic model, a level of indenture, and a group size *n*, and it returns the set of diagnosis ambiguity groups (containing only diagnoses of type "failure") of exactly size *n*. An additional BOOLEAN argument is provided as input to indicate whether to include ambiguity with No Fault in the analysis. The model specified must be a DIM, a FTM, or an EDIM.

```
FUNCTION isolatable_failures_set
   (mdl : diagnostic_model;
    lvl : level;
    n : INTEGER;
    include_nf : BOOLEAN) : SET OF diagnosis_ambiguity;
END_FUNCTION;
```

5.4.18 isolatable_faults_set

The fundamental function `isolatable_faults_set` takes as input a diagnostic model, a level of indenture, and a group size *n*, and it returns the set of diagnosis ambiguity groups (containing only diagnoses of type "fault") of exactly size *n*. An additional BOOLEAN argument is provided as input to indicate whether to include ambiguity with No Fault in the analysis. The model specified must be a DIM, a FTM, or an EDIM.

```
FUNCTION isolatable_faults_set
   (mdl : diagnostic_model;
    lvl : level;
    n : INTEGER;
    include_nf : BOOLEAN) : SET OF diagnosis_ambiguity;
END_FUNCTION;
```

5.4.19 fault_ambig_set

The fundamental function `fault_ambig_set` takes as input a diagnostic model, a level of indenture, and a group size *n*, and it returns the set of diagnosis ambiguity groups (containing only diagnoses of type "fault") for which the constituent faults are associated with exactly *n* repair items. An additional BOOLEAN argument is provided as input to indicate whether to include ambiguity with No Fault in the analysis. The model specified must be a DIM, a FTM, or an EDIM.

```
FUNCTION fault_ambig_set
   (mdl : diagnostic_model;
    lvl : level;
    n : INTEGER;
    include_nf : BOOLEAN) : SET OF diagnosis_ambiguity;
END_FUNCTION;
```

5.4.20 failure_ambig_set

The fundamental function `failure_ambig_set` takes as input a diagnostic model, a level of indenture, and a group size *n*, and it returns the set of diagnosis ambiguity groups (containing only diagnoses of type "failure") for which the constituent failures are associated with exactly *n* repair items. An additional BOOLEAN argument is provided as input to indicate whether to include ambiguity with No Fault in the analysis. The model specified must be a DIM, a FTM, or an EDIM.

```
FUNCTION failure_ambig_set
   (mdl : diagnostic_model;
    lvl : level;
    n : INTEGER;
    include_nf : BOOLEAN) : SET OF diagnosis_ambiguity;
END_FUNCTION;
```

6. Metrics

During development of this standard, it became obvious that it would be impractical to delineate all possible metrics. Metrics can be tailored to answer specific questions about the testability of a system, and it is not the intent of this standard to exclude worthwhile metrics. Instead of a voluminous list of metrics, a series of building blocks (the fundamentals) was developed. All fundamentals are based on IEEE Std 1232-2002 (AIESTATE). The metrics in this clause all use these fundamentals. Several metrics are included as informative, rather than as normative, to illustrate the process of developing a metric. Metrics developed using the fundamentals can be submitted to the D&MC Subcommittee for possible inclusion in a later revision of this standard.

6.1 Fault detection metrics

This subclause provides definitions of the testability and diagnosability metrics that quantify the fault detection capability of a given diagnostic model. For each fault detection metric, an equation is provided that describes that metric. There are two detection metrics—one that describes a diagnostic model's fault detection capability as an unweighted percentage (percentage of detection) and one that uses a percentage weighted by probability of diagnosis (expected percentage of detection).

6.1.1 Percentage of detection

Percentage of detection (*PoD*) is the percentage of possible diagnoses at a particular level within a specified diagnostic model that can be detected by a given set of tests that have been defined within that model. This metric is described by Equation (1):

$$PoD(mdl,lvl) = \frac{100 \times |\texttt{single\_level\_detection}(ids,lvl)|}{|\texttt{diagnosis\_set}(mdl,lvl)|} \quad (1)$$

where

ids is the `implicated_diagnosis_set (mdl, lvl)`
mdl is the diagnostic model for which this metric is to be calculated
lvl is the level of the diagnostic model at which diagnoses are to be counted

6.1.2 Expected percentage of detection

Expected percentage of detection (*EPoD*) is the failure probability-weighted percentage of possible diagnoses at a particular level within a specified diagnostic model that can be detected by a given set of tests that have been defined within that model. This metric is described by Equation (2):

$$EPoD(mdl,lvl) = 100 \times \left[\sum_{\forall D_i \in isld} \text{probability\_of\_diagnosis}(D_i,sld) \right] \quad (2)$$

where

D_i is the *i*th diagnosis in *isld*
isld is the set returned by `single_level_detection`(*ids, lvl*)
ids is `implicated_diagnosis_set`(*mdl, lvl*)
sld is the set returned by `single_level_detection`(*ds, lvl*)
ds is `diagnosis_set`(*mdl, lvl*)
mdl is the diagnostic model for which this metric is to be calculated
lvl is the level of the diagnostic model at which diagnoses are to be counted

6.2 Fault isolation metrics

This subclause provides definitions of the testability and diagnosability metrics that quantify the fault isolation capability of a given diagnostic model. For each fault isolation metric, an equation is provided that describes that metric. There are six isolation metrics: percentage of isolation, expected percentage of isolation, expected ambiguity group size, isolation effectiveness, projected effect of ambiguity upon false removals, and repair item involvement ratio. The first and sixth of these metrics (percentage of isolation and repair item involvement ratio) are unweighted, whereas the other four metrics are weighted using probability of diagnosis.

6.2.1 Percentage of isolation

Percentage of isolation consists of two individual equations—the incremental percentage of fault isolation and the cumulative percentage of isolation. The incremental percentage of isolation (*IPoI*) is, for a particular level of a specified model, the percentage of all diagnosis ambiguity groups that can be isolated using the tests defined within that model and that contain a specified number of repair items. This metric is described by Equation (3):

$$IPoI(mdl,lvl,n) = \frac{100 \times |\texttt{fault\_ambig\_set}(mdl,lvl,n)|}{\sum_{\forall size} |\text{fault\_ambig\_set}(mdl,lvl,size)|} \tag{3}$$

where

mdl	is the diagnostic model for which this metric is to be calculated
lvl	is the level of the diagnostic model at which faults are to be counted
n	is the specified number of repair items
size	is the maximum size of an ambiguity group in the model

Cumulative percentage of isolation (*CPoI*) is, for a particular level of a specified diagnostic model, the percentage of all diagnosis ambiguity groups that can be isolated using the tests defined within that model and that contain *no more than* a specified number of repair items. This metric is described by Equation (4):

$$CPoI(mdl,lvl,n) = \sum_{i=1}^{n} IPoI(mdl,lvl,i) \tag{4}$$

where

mdl	is the diagnostic model for which this metric is to be calculated
lvl	is the level of the diagnostic model at which faults are to be counted
n	is the specified number of repair items

6.2.2 Expected percentage of isolation

Expected percentage of isolation consists of two individual equations—the incremental expected percentage of isolation and the cumulative expected percentage of isolation. The incremental expected percentage of isolation (*IEPoI*) is, for a particular level of a specified diagnostic model, the failure probability-weighted percentage of all diagnosis ambiguity groups that can be isolated using the tests defined within that model and that contain a specified number of repair items. This metric is described by Equation (5):

$$IEPoI(mdl,lvl,n) = 100 \times \sum_{\forall D_i \in AG_n} P(D_i) \tag{5}$$

where

AG_n	is the UNION(RD_n) (i.e., the set union)
RD_n	is the `convert_repair_to_diag_ambiguity`(*mdl, lvl,* RAG_n)
RAG_n	is the `get_size_n_ambig_group`(*mdl, lvl, n*)
D_i	is the *i*th diagnosis in AG_n
$P(D_i)$	is the `probability_of_diagnosis`(D_i, *isld*)
isld	is the the set returned by `single_level_detection`(*ids, lvl*)
ids	is the `implicated_diagnosis_set`(*mdl, lvl*)
mdl	is the diagnostic model for which this metric is to be calculated
lvl	is the level of the diagnostic model at which faults are to be counted

n is the specified number of repair items

When calculating *IEPoI*, all repair item ambiguity groups of size *n* must first be found. The probability of isolating to one ambiguity group is equal to the sum of the probabilities of the diagnoses associated with the repair items that caused the ambiguity. Therefore, the incremental expected probability of isolating to one ambiguity group is the sum of the probabilities of all diagnoses causing the collection of repair item ambiguity groups. *IEPoI* will then provide the probability of isolating to a repair item ambiguity group exactly of size *n*.

Cumulative expected percentage of isolation (*CEPoI*) is, for a particular level of a given diagnostic model, the failure probability-weighted percentage of all diagnosis ambiguity groups that can be isolated using the tests defined within that model and that contain *no more than* a specified number of repair items. In other words, *CEPoI* provides the probability of isolating to a repair item ambiguity group of size *n* or less. This metric is described by Equation (6):

$$CEPoI(mdl,lvl,n) = \sum_{i=1}^{n} IEPoI(mdl,lvl,i) \tag{6}$$

where

mdl is the diagnostic model for which this metric is to be calculated
lvl is the level of the diagnostic model at which faults are to be counted
n is the specified number of repair items

6.2.3 Expected ambiguity group size

Expected ambiguity group size (*EAGS*) is, for a particular level of a given diagnostic model, the failure probability-weighted average number of repair items in all repair item ambiguity groups that can be isolated using the tests defined within that model. This metric is described by Equation (7):

$$EAGS(mdl,lvl) = \sum_{i=1}^{n} i \times IEPoI(mdl,lvl,i) \tag{7}$$

where

mdl is the diagnostic model for which this metric is to be calculated
lvl is the level of the diagnostic model at which faults are to be counted
n is the maximum number of repair items in an ambiguity group in the model

6.2.4 Isolation effectiveness (informative)

This sublcause is informative only and is not a normative part of this standard.

Isolation effectiveness (IE) is, for a particular level of a given diagnostic model, a measure of how well the repair item ambiguity groups that can be isolated using tests defined within that model conform to a specified diagnostic goal (expressed as a lower bound/minimum). Typical goals include the ambiguity group size or the cost/time to isolate, replace or repair ambiguity groups.

Let $\Delta(f(x), y) = \begin{cases} f(x) - y & f(x) \geq y \\ 0 & otherwise \end{cases}$ be the deviation of a function of *x* relative to the goal *y*.

Then the *general form* of this metric is described by Equation (8):

$$IE(mld,lvl,goal) = 1 - \frac{\sum_{i=1}^{n} P(AG_i) \times \Delta(f(AG_i),goal)}{\sum_{i=1}^{n} P(AG_i) \times f(AG_i)} \quad (8)$$

where

AG_i	is the ith repair item ambiguity group
goal	is the goal (expressed as a lower bound/minimum) to which the calculated value for each ambiguity group is to be compared (such as the target ambiguity group size or repair cost)
$f(AG_i)$	is the value of the measure for AG_i that is to be compared with the specified goal (such as the actual ambiguity group size or repair time for AG_i)
$P(AG_i)$	is the `probability_of_repair_item_set`(AG_i), i.e., the probability that AG_i is isolated
mdl	is the diagnostic model for which this metric is to be calculated
lvl	is the level of the diagnostic model at which faults are to be counted
n	is the total number of repair item ambiguity groups for the *lvl* and *mdl*

Note that when all repair item ambiguity groups meet the specified goal, $((f(AG_i),goal)$ will be equal to zero for all ambiguity groups and the isolation effectiveness metric will calculate to a value of 1.0 (100% effectiveness).

For example, one of the more common uses of the isolation effectiveness metric is as an indication of how well a given diagnostic model can achieve isolation to a maximum size (e.g., a single failed repair item). For this usage, let $x = AG_i$. Then $f(AG_i) = |AG_i|$ (i.e., the size of repair item ambiguity group AG_i). Then let $y = size$, which thus specifies the maximum size. Thus, the specific form of the isolation effectiveness metric, which is denoted IE_{isol}, becomes:

$$IE_{isol}(mdl,lvl,size) = 1 - \frac{\sum_{i=1}^{n} P(AG_i) \times \Delta(|AG_i|,size)}{\sum_{i=1}^{n} P(AG_i) \times |AG_i|}$$

where

AG_i	is the *i*th repair item ambiguity group
size	is the goal specifying the maximum size of an ambiguity group
$\|AG_i\|$	is the actual size of AG_i
$P(AG_i)$	is the `probability_of_repair_item_set`(AG_i), i.e., the probability that AG_i is isolated
mdl	is the diagnostic model for which this metric is to be calculated
lvl	is the level of the diagnostic model at which faults are to be counted
n	is the total number of repair item ambiguity groups for the *lvl* and *mdl*

In this example, setting *size* = 1 will test the ability to isolate to a single repair item.

As an additional example, isolation effectiveness can be used to identify the effect of diagnostic ambiguity on a particular maintenance goal such as maximum repair time. Again, letting $x = AG_i$ and defining $f(AG_i)$ to

be the repair time associated with AG_i (denoted $RT(AG_i)$, Equation (9) gives this variant of IE (denoted IE_{RT}).

$$IE_{RT}(mdl,lvl,time) = 1 - \frac{\sum_{i=1}^{n} P(AG_i) \times \Delta(RT(AG_i),time)}{\sum_{i=1}^{n} P(AG_i) \times RT(AG_i)} \quad (9)$$

where

AG_i	is the *i*th repair item ambiguity group
time	is the goal specifying the maximum repair time of an ambiguity group
$RT(AG_i)$	is the actual repair time of AG_i
$P(AG_i)$	is the `probability_of_repair_item_set`(AG_i), i.e., the probability that AG_i is isolated
mdl	is the diagnostic model for which this metric is to be calculated
lvl	is the level of the diagnostic model at which faults are to be counted
n	is the total number of repair item ambiguity groups for the *lvl* and *mdl*

To illustrate, suppose *time* = 60. In this case, the delta function returns zero if the repair time for an ambiguity group is less than the specified goal of 60 minutes. An alternative version of this metric could penalize for exceeding the goal by too much (rather than returning zero when the goal is exceeded, the delta function could return the absolute value of the difference).

6.2.5 Projected effect of ambiguity on false removals (informative)

This subclause is informative only and is not a normative part of this standard.

Projected effect of ambiguity on false removals (PEAFR) is, for a particular level of a given diagnostic model, a measure of the effect that ambiguous isolation has on the frequency of false removals, given the set of repair item ambiguity groups that can be isolated using tests defined within that model. This metric is described by Equation (10):

$$PEAFR(mdl,lvl) = \frac{\sum_{i=1}^{n} P(AG_i) \times (|AG_i| - 1)}{\sum_{i=1}^{n} P(AG_i) \times (|AG_i|)} \quad (10)$$

where

AG_i	is the *i*th repair item ambiguity group
$\|AG_i\|$	is the size of AG_i
$P(AG_i)$	is the `probability_of_repair_item_set`(AG_i), i.e., the probability that AG_i is isolated
mdl	is the diagnostic model for which this metric is to be calculated
lvl	is the level of the diagnostic model at which faults are to be counted
n	is the total number of repair item ambiguity groups for the *lvl* and *mdl*

PEAFR gives the worst-case percentage of false removals because of diagnostic ambiguity for the specified level of the given diagnostic model. It projects the relative likelihood of false removals when malfunctions

occur singly and repair item ambiguity groups are replaced as a block. If isolated ambiguity groups contain multiple faulty repair items or if ambiguity groups are maintained using serial (prioritized) replacement, then the actual percentage of false removals would be lower. During maintenance actions, false removals can result from factors other than ineffective fault isolation—PEAFR only quantifies the effect of diagnostic ambiguity on false removals.

Note that PEAFR is equal to 1 minus IE with respect to isolation to a single failed repair item. In other words, $PEFR(mdl,lvl) = 1 - IE_{isol}(mdl, lvl, 1)$.

6.2.6 Repair item involvement ratio

Repair item involvement ratio (RIIR) is, for a particular level of a given diagnostic model, the number of repair item ambiguity groups that contain a given repair item divided by the total number of ambiguity groups at that level that can be isolated using tests defined within that model. This metric is described by Equation (11):

$$RIIR(mdl,lvl, RI) = \frac{|AG_{RI}|}{\bigcup_{\forall RI} AG_{RI}} \tag{11}$$

where

AG_{RI}	is the `get_ambig_group`(*mdl, RI, lvl*)
$\|AG_{RI}\|$	is the size of AG_{RI}
RI	is the repair item for which the *RIIR* is being calculated
mdl	is the diagnostic model for which this metric is to be calculated
lvl	is the level of the diagnostic model at which faults are to be counted

6.3 Variants and constraints

This subclause is informative only and is not a normative part of this standard.

6.3.1 Functional subsets

Fault detection and isolation metrics can be calculated over an entire design or over an explicitly defined functional subset of that design. The following list provides some typical types of subsets that may be used (individually or in combination) to constrain the fault detection metrics for a design:

- Functionality (e.g., avionics, weapon systems, survivability)
- Technology (e.g., electronic, mechanical, hydraulic)
- Operational usage (operational hardware vs. built-in test (BIT) hardware/sensors)
- Hardware source (e.g., customer-furnished equipment, commercial off-the-shelf)
- Operating mode (e.g., takeoff, autopilot, on-line, off-line, in storage)
- Test domain (e.g., hardware functions intended to be tested by startup BIT)
- Fault universe (specific fault types over which detection is to be calculated)

When calculating the metrics for a particular functional subset of a design, the equations defined in this clause can be used, provided that the model (*mdl*) is constructed such that only the desired subset is represented.

6.3.2 Classes of tests

Fault detection and isolation metrics are frequently constrained by test class to calculate the fault coverage associated with different subsets of tests within the design. Common test classes that are used to constrain analysis include

— BIT type (e.g., startup, continuous, periodic)
— Diagnostic usage (e.g., embedded diagnostics, technical manual tests)
— Test implementations (e.g., current BIT, proposed BIT, proposed technical manual tests)

When calculating the metrics for a particular subset of tests within a design, the equations defined in this clause can be used, provided that the model (*mdl*) is constructed such that only the desired subset is represented.

6.3.3 Time interval

The diagnosis failure probabilities used to calculate the *EPoD* and *EPoI* are generally derived from failure rates—statistical means that describe expected rates of failure, usually over an arbitrarily long time interval (these means are most frequently expressed in terms of failures per million hours). The same metric equation can be used to calculate *EPoD* and *EPoI* over a specific time interval (such as the first two years of deployment), provided that all failure rates in the specified diagnostic model (*mdl*) represent the number of failures expected within the given time interval (rather than the number of failure expected per million hours).

Annex A

(informative)

Bibliography

[B1] DePaul, R. A. Jr., "Logic modeling as a tool for testability," *Proceedings of the IEEE AUTOTESTCON*. New York: IEEE Press, 1985, pp. 203–207.

[B2] Gould, E., "Serial replacement maintenance philosophies and multiple-failure diagnostic strategies: A marriage of multiple-fault integrity and common cause sensibility," *Proceedings of the IEEE AUTOTESTCON*. New York: IEEE Press, 1997 pp. 446–454.

[B3] Gould, E., and Hartop, D., "Thinking beyond the group size fetish: Towards a new testability," *Proceedings of the IEEE AUTOTESTCON*. New York: IEEE Press, 1999, pp. 673–684.

[B4] IEEE 100, The Authoritative Dictionary of IEEE Standards Terms.[6]

[B5] Keiner, W., "A Navy approach to integrated diagnostics," *Proceedings of the IEEE AUTOTESTCON*. New York: IEEE Press, 1990, pp. 443–450.

[B6] Naval Sea Systems Command, *Testability Analysis Handbook*, 1989.

[B7] Qualtech Systems Inc., *Teams: Testability Engineering and Maintenance System Users Guide 4.1*, 1996.

[B8] Simpson, W. R., Bailey, J., Barto, K., and Esker, E., "Organizational-level testability prediction," U.S. Air Force, Apr. 1985.

[B9] Simpson, W, R., and Sheppard, J. W., *System Test and Diagnosis*. Norwell, MA: Kluwer Academic, 1994.

[B10] U.S. Navy, *Definitions of Terms for Test, Measurement and Diagnostic Equipment*, MIL-STD-1309C, Naval Electronics Systems Command (ELEX-8111), Washington, D.C., Nov. 1983.

[B11] U.S. Navy, *Testability Program for Electronic Systems and Equipment*, MIL-STD-2165, Naval Electronics Systems Command (ELEX-8111), Washington, D.C., Jan. 1985.

[6]IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA (http://standards.ieee.org/).

Annex B

(informative)

Product life cycle and metric applicability

It is often helpful to look at concepts such as the metrics and characteristics of this specification in an applicable context within the domain of discourse. That domain in this case is inarguably that of diagnostic design and development, which is, in most cases, an integral part of the system engineering process as applied to product design, development, and use.

This annex describes one such usage scenario for the information elements defined herein. The scenario represented is the product life cycle process as it relates to diagnostic analysis and design.

A comprehensive diagnostic analysis and design program requires a dedicated effort throughout all phases of the system life cycle to assure that the end item will satisfy all customer needs and expectations relative to its diagnostic performance. The diagnostic design life cycle is organized around the five-system life cycle phases:

— Requirements definition

— Architecture design

— Detailed test design

— Verification and validation

— Diagnostic maturation

A brief description of each phase is provided in B.1–B.7.

B.1 Requirements definition

The first step in an effective testability program is to define a complete set of requirements that can be effectively used to manage diagnostic analysis and development and assure that the product will satisfy all customer needs and expectations in this regard. Diagnostic requirements are typically expressed in performance terms. It is important that they express what the desired outcome is, without directing how to achieve that outcome. Goals are typically specified that center around fault detection (FD), fault isolation (FI), and fault isolation resolution (FIR) or operational isolation. A related requirement that is examined early in the design phase of a program is mission reliability. Although it has typically been used as a verification of the reliability of systems that can be reconfigured, redundant, or safety-critical, it is being applied more frequently as a measure of the prognostic capability of a system to satisfy mission requirements with a satisfactory measure of confidence; i.e., the system prognostics are shown analytically to be capable of predicting some interval over which the system will operate without malfunction. Other system-level metrics that may be specified at this time are mean time between failure (MTBF), operational availability, can not duplicate (CND)/no evidence of failure (NEOF) rate, and mean time between unscheduled maintenance actions (MTBUMA). Derivative requirements include mean time to repair (MTTR), maintenance frequency, skill levels of repair personnel, and time required for maintenance, man-hours/flight hours, turnaround time, fault isolation time, and repair and replace time. Furthermore, customer warranties may be executed on some metrics such as MTBF or operational availability or even on total operating hours.

B.2 Architecture design

The intent in the architecture design phase is to develop a preliminary design framework that allows all diagnostic program requirements to be allocated in a top-down manner to the physical system components. A basic diagnostic subsystem architecture is laid out in conjunction with any related trade studies (e.g., fault tolerance, fault reporting, field update approaches, prognostic vs. diagnostic approaches, etc.), and the evaluation of the system's fault tolerance of alternative architectures against cost, weight, power, and MTBF or mean time between critical failure (MTBCF). Subsystem requirements are then allocated to individual hardware elements (typically boxes and/or modules) to drive detailed design requirements.

System-level testability models may be created at this time to support this analysis, perhaps in conjunction with a functional failure modes, effects and criticality analysis (FMECA). These models should evolve with the design. Testability analysis is performed on the models of the design in a top-down manner to verify that the functional/physical partitioning and diagnostic requirements allocation will realistically satisfy the top level requirements, using the same metrics as identified in B.1. At the same time, the analysis serves to provide guidance for diagnostic design influence (the process of assuring that the diagnostic design features necessary to meet testability requirements are identified and incorporated into the product design). It is typically during this process that an elementary framework for a test strategy begins to emerge.

B.3 Detailed test design

The purpose of the detailed test design phase is to develop a detailed design solution that meets all diagnostic program requirements. During this phase, detailed hardware and software designs are created in accordance with allocated system requirements. As the physical design matures, the hardware aspects are reflected in the analysis models and linked to their functional elements. The basic diagnostic subsystem architecture is also modified as design and support system details emerge. Diagnostic and reliability trade studies (e.g., fault tolerance, fault reporting, prognostics and field update approaches) and their effect on cost, weight, power, MTBCF, and MTBF are iterated as required, and allocations of FD, FI, and FIR are adjusted as necessary. The results of these modifications are reflected in the models and verified.

It is during this phase that preliminary test verticality (cone of tolerance analysis) is performed. The results of this analysis are the comparative margins and tolerances of the tests at the various levels of test, which may include

— BIT vs. automatic test equipment (ATE) test consistency

— Factory vs. field test consistency

— Worst-case tolerance variations

— Latent manufacturing defects

B.4 Verification and validation

The purpose of the verification and validation phase is to validate that the detailed design solution, which was verified by analysis to assure compliance with specification requirements, will actually achieve all diagnostic requirements on delivery. This approach uses test and demonstration to verify that each built-in test operates correctly. In addition, diagnostic support equipment and any support system contributions to diagnostics are tested or demonstrated as part of operational and maintenance scenarios. Quantitative testability allocations and the inherent design requirements are verified through analysis and inspection methods.

During this phase, two primary validation efforts are performed. First, the validity of the fault detection and fault isolation models are assessed through a static *fault insertion* (e.g., BIT demonstration). Second, susceptibility to false alarms and can not duplicates is evaluated during field testing opportunities.

B.5 Fault insertion

As diagnostic software only performs its intended job in the presence of failures, a fault insertion (e.g., BIT demonstration) is the only controlled opportunity for exercising the complete embedded hardware/software design. Typically, this result is accomplished by physically inserting a limited number of nondestructive faults into the hardware design and verifying that they are properly detected and reported. The actual diagnostic software must be executed to verify that the tests assumed during the analysis phases were effectively implemented in software.

Faults selected for insertion should be chosen based on highest failure probability, maximum amount of diagnostic software exercised, and least potential damage to the unit under test.

Corrective actions for FD/FI deficiencies are to be implemented and validated.

B.6 False alarm and can not duplicate susceptibility

False alarms and can not duplicates are undesirable design characteristics that cannot truly be predicted or systematically tested, but some risk reduction activities can be performed in conjunction with other activities to mitigate the risk incurred by the possibility of their occurrence. Examples of such opportunities are as follows.

Development and qualification tests: As these tests exercise the hardware under simulated environmental and life conditions, it is possible that anomalies can be corrected in a “test, analyze, and fix” methodology even though specification compliance cannot truly be evaluated.

Laboratory and field tests: As these tests are performed using simulated or actual interfaces, they provide an opportunity to correct diagnostic anomalies.

B.7 Diagnostic maturation

The purpose of the maturation phase is to implement a program that effectively measures actual product performance and to identify/implement corrective actions as required. The effort to “mature” a design begins at the conceptual design stage and continues throughout the system life cycle. A maturation plan must be constructed to document how data indicative of diagnostics performance within actual operational and maintenance environments will be collected, measured, evaluated, and acted on.

At the completion of the validation phase, the state of the diagnostic design will be demonstrated to meet the diagnostic and health management requirements that have evolved before delivery of the first unit. However, once a system is fielded and starts to be used in an operational environment, unexpected and unplanned system-level design interactions, operational and environmental stresses, and other influences can degrade the performance of the diagnostic design from what was predicted. When this degradation results in unacceptable product performance or other cost of ownership problems, remedial actions must be taken. Root cause analysis must be performed to identify potential adjustments, improvements, or refinements to diagnostic, support, or operational elements.

Annex C

(normative)

Extensions to the AI-ESTATE standard

This annex contains the specification for extensions of the AI-ESTATE information models that are required by the `probability_of_diagnosis` and `probability_of_repair_item_set` measures that are defined within this document.

```
(* EXPRESS Specification starts here. *)
(*
```

C.1 The AI-ESTATE 1522 extension

This extension redefines the diagnosis entity in the common element model defined within IEEE Std 1232-2002 (AI-ESTATE), which allows a failure rate value to be assigned to each diagnosis.

Express specification:

```
*)
SCHEMA EXTEND_1522;
USE FROM AI_ESTATE_COMMON_ELEMENT_MODEL
      (diagnosis, failure_rate, frequency);
(*
```

C.1.1 EXTEND_failure_rate

The failure rate extension (`EXTEND_failure_rate`) is an abstract entity that is a subtype of the `failure_rate` entity in the AI-ESTATE common element model and contains the frequency value for the associated failure rate. There are two subtypes for this entity: `EXTEND_weibull_failure_rate` and `EXTEND_exponential_failure_rate`.

Express specification:

```
*)
ENTITY EXTEND_failure_rate
ABSTRACT SUPERTYPE OF
      (ONE OF(EXTEND_weibull_failure_rate,
        EXTEND_exponential_failure_rate))
SUBTYPE OF (failure_rate);
associated_frequency : frequency;
END_ENTITY;
(*
```

Attribute definitions:

associated_frequency : Attribute used to specify the rate (number of occurrences) of failures that are associated with a diagnosis over a given time span.

C.1.2 EXTEND_weibull_failure_rate

The entity EXTEND_weibull_failure_rate is a subtype of the abstract entity EXTEND_failure_rate that is used when the associated failure rate is the result of a Weibull distribution.

Express specification:

```
*)
ENTITY EXTEND_weibull_failure_rate
SUBTYPE OF (EXTEND_failure_rate);
alpha : REAL;
beta : REAL;
END_ENTITY;

(*
```

Attribute definitions:

alpha : Attribute used to specify the alpha parameter of the Weibull distribution associated with the given failure rate.

beta : Attribute used to specify the beta parameter of the Weibull distribution associated with the given failure rate.

C.1.3 EXTEND_exponential_failure_rate

The entity EXTEND_exponential_failure_rate is a subtype of the abstract entity EXTEND_failure_rate that is used when the associated failure rate is the result of an exponential distribution.

Express specification:

```
*)
```

```
ENTITY EXTEND_exponential_failure_rate
SUBTYPE OF (EXTEND_failure_rate);
END_ENTITY;

(*
```

C.1.4 EXTEND_diagnosis

The entity `EXTEND_diagnosis` is a subtype of the diagnosis entity in the AI-ESTATE common element model and contains the failure rate for the associated diagnosis.

```
Express specification:

*)

ENTITY EXTEND_diagnosis
SUBTYPE OF (diagnosis);
new_rate : EXTEND_failure_rate;
END_ENTITY;

(*

Attribute definitions:

   new_rate  : Attribute used to specify the failure rate associated with the given diagnosis.
*)

END_SCHEMA;
```