IEEE Standard Classification for Software Anomalies

IEEE Computer Society

Sponsored by the
Software & Systems Engineering Standards Committee

1044™

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

7 January 2010

IEEE Std 1044™-2009
(Revision of
IEEE Std 1044-1993)

IEEE Std 1044™-2009
(Revision of
IEEE Std 1044-1993)

IEEE Standard Classification for Software Anomalies

Sponsor

Software & Systems Engineering Standards Committee

of the

IEEE Computer Society

Approved 9 November 2009

IEEE-SA Standards Board

Abstract: This standard provides a uniform approach to the classification of software anomalies, regardless of when they originate or when they are encountered within the project, product, or system life cycle. Classification data can be used for a variety of purposes, including defect causal analysis, project management, and software process improvement (e.g., to reduce the likelihood of defect insertion and/or to increase the likelihood of early defect detection).
Keywords: anomaly, bug, classification, defect, error, failure, fault, problem

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

PDF: ISBN 978-0-7381-6114-3 STD95995
Print: ISBN 978-0-7381-6115-0 STDPD95995

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 or the soundness of any judgments 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, or every ten years for stabilization. When a document is more than five years old and has not been reaffirmed, or more than ten years old and has not been stabilized, 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 his or her independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

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. A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered the official position of IEEE or any of its committees and shall not be considered to be, nor be relied upon as, a formal interpretation of the IEEE. 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. Recommendations to change the status of a stabilized standard should include a rationale as to why a revision or withdrawal is required. Comments and recommendations on standards, and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854
USA

Introduction

This introduction is not part of IEEE Std 1044-2009, IEEE Standard Classification for Software Anomalies.

This standard provides a uniform approach to the classification of software anomalies, regardless of when they originate or when they are encountered within the project, product, or system life cycle. Classification data can be used for a variety of purposes, including defect causal analysis, project management, and software process improvement (e.g., to reduce the likelihood of defect insertion and/or to increase the likelihood of early defect detection).

Collecting the data described in this standard provides valuable information that has many useful applications. It is also well documented that the earlier within the software life cycle a problem is discovered, the cheaper, and often easier, it is to fix. This encourages the use of tools, techniques, and methodologies to find problems sooner. Standard anomaly data are necessary to evaluate how well these tools, techniques, and methodologies work. These data can also identify when in a project's life cycle most problems are introduced. Distinctions between enhancements and problems in the software help make the decisions as to which anomalies are addressed first, categories of funding, and so on. Anomaly data can also assist in the evaluation of quality attributes such as reliability and productivity.

Having a standard way of classifying software anomalies is important. First, it enables insight into the types of anomalies that organizations produce during development of their products. This information is a rich source of data that can be used during the execution of a project and for process improvement. Analytical techniques such as Orthogonal Defect Classification and causal analysis depend on classification of anomalies to identify root causes and to help determine means to prevent their recurrence. Process improvement frameworks such as Capability Maturity Model Integrated® (CMMI®)[a] promote the need for detailed understanding of process performance and product quality. The classification of anomalies allows the development of profiles of anomalies produced by various development processes as one indicator of process performance.

Second, having a standard way to classify anomalies enables better communication and exchange of information regarding anomalies among developers and organizations. Unfortunately, people frequently associate different meanings with the same words and/or use different words to mean the same thing. Similarly, if software systems are to communicate (i.e., exchange data) effectively regarding anomalies, they must share a common logical (if not physical) data model. Data exchange may still be possible via some mapping or translation method if the same data elements are named differently in one system as compared with another, but each system must at least recognize and implement the same conceptual objects/entities, relationships, and attributes.

This standard is based on several concepts and definitions that must be clearly understood prior to its use. These are discussed briefly in the following paragraphs. Formal definitions can be found in Clause 2, and it is advisable to read them before proceeding.

The word "anomaly" may be used to refer to any abnormality, irregularity, inconsistency, or variance from expectations. It may be used to refer to a condition or an event, to an appearance or a behavior, to a form or a function. The 1993 version of IEEE Std 1044™ characterized the term "anomaly" as a synonym for error, fault, failure, incident, flaw, problem, gripe, glitch, defect, or bug, essentially deemphasizing any distinction among those words. Such usage may be common practice in everyday conversation where the inherent ambiguity is mitigated by the richness of direct person-to-person communication, but it is not conducive to effective communication by other (especially asynchronous) methods. Because a term with such a broad meaning does not lend itself to precise communication, more specific terms are defined and used herein to refer to several more narrowly defined entities. Each entity has associated with it a name, a

[a] CMMI and Capability Maturity Model Integrated are registered trademarks in the U.S. Patent & Trademark Office, owned by the Carnegie Mellon Software Engineering Institute. This information is provided for the convenience of users of this standard and does not constitute an endorsement by the IEEE of these products.

definition, a set of attributes, and a set of relationships to other entities. These entities are depicted in Figure 1 and again (using a different notation) in Figure 2, and their definitions can be found in Table 3. The relationships between key entities are modeled in the form of an entity relationship diagram (ERD) in Figure 1 and again in the form of a Unified Modeling Language (UML®)[b] class diagram in Figure 2. A detailed explanation of these widely used modeling notations is outside the scope of this standard; however, a brief description of each is provided below the corresponding diagram. Text descriptions of the relationships are available in Table 1, so a complete understanding of the diagrams is not essential to understanding this standard. The entities "Failure," "Defect," and "Fault" within the area labeled "IEEE 1044 Scope" in Figure 1 and Figure 2 are the subject of this standard, and the other entities in the diagrams are outside the scope of this standard. Faults are considered a subtype of defect and as such are classified using the same attributes as defects (see Table 4).

To increase flexibility and allow organizations to adapt the classification to their own life cycles and purposes, the following changes have been made to the previous edition of this standard:

— Defining key terms and the relationships between their underlying concepts more precisely

— Not specifying a mandatory set of values for anomaly attributes

— Not specifying a classification process

Several concepts and definitions must be clearly understood before using this standard, so it is highly advisable to review 1.1 and Clause 2 carefully before proceeding to Clause 3. The classification attributes defined in the standard are normative (mandatory), whereas the sample classification attribute values are only informative (optional).

Notice to users

Laws and regulations

Users of these documents should consult all applicable laws and regulations. Compliance with the provisions of this standard does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

Updating of IEEE documents

Users of IEEE standards should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect. In order to determine whether a given document is the current edition and whether it has been amended through the issuance of

[b] UML is a registered trademark in the U.S. Patent & Trademark Office, owned by the Object Management Group.

amendments, corrigenda, or errata, visit the IEEE Standards Association Web site at http://ieeexplore.ieee.org/xpl/standards.jsp, or contact the IEEE at the address listed previously.

For more information about the IEEE Standards Association or the IEEE standards development process, visit the IEEE-SA Web site at http://standards.ieee.org.

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 is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Participants

At the time this standard was submitted to the IEEE-SA Standards Board for approval, the IEEE 1044 Working Group had the following membership:

David Zubrow, *Chair*
Mark Baldwin, *Vice Chair*

Maryam Asdjodi-Mohadjer
Karen Butler
David Card
Robert Douglas
John Gaffney
Mark Hadley
Jim Kubeck
Celia Modell
Sharon Rohde
Virginia Slavin

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

Ed Addario
Johann Amsenga
Butch Anton
Angela Anuszewski
Chris Bagge
Bakul Banerjee
Zulema Belyeu
H. Stephen Berger
Juris Borzovs
Pieter Botman
Juan Carreon
Keith Chow
Daniel Conte
Paul Croll
Geoffrey Darnton
Giuseppe De Francesco
Terry Dietz
Thomas Dineen
Antonio Doria
Richard Doyle
Edward Dudash
Scott Duncan
Sourav Dutta
Carla Ewart
Yaacov Fenster
Andrew Fieldsend
Andre Fournier
David Friscia
Gregg Giesler
Randall Groves
John Harauz
Mark Henley
Rutger A. Heunks
Werner Hoelzl
Atsushi Ito
Mark Jaeger
Cheryl Jones
Piotr Karocki
Rameshchandra Ketharaju
Dwayne Knirk
Ronald Kohl
Thomas Kurihara
George Kyle
Susan Land
Dewitt Latimer
David J. Leciston
Daniel Lindberg
Vincent Lipsio
Carol Long
Faramarz Maghsoodlou
Edward McCall
William Milam
James Moore
Mark Paulk
Miroslav Pavlovic
Ulrich Pohl
Annette Reilly
Robert Robinson
Terence Rout
Randall Safier
Bartien Sayogo
Robert Schaaf
Hans Schaefer
David J. Schultz
Stephen Schwarm
Gil Shultz
James Sivak
Thomas Starai
Walter Struppler
Gerald Stueve
Marcy Stutzman
Thomas Tullia
Vincent Tume
John Walz
P. Wolfgang
Paul Work
Oren Yuen
Janusz Zalewski

When the IEEE-SA Standards Board approved this standard on 9 November 2009, it had the following membership:

Robert M. Grow, *Chair*
Thomas Prevost, *Vice Chair*
Steve M. Mills, *Past Chair*
Judith Gorman, *Secretary*

John Barr
Karen Bartleson
Victor Berman
Ted Burse
Richard DeBlasio
Andy Drozd
Mark Epstein
Alexander Gelman
Jim Hughes
Richard H. Hulett
Young Kyun Kim
Joseph L. Koepfinger*
John Kulick
David J. Law
Ted Olsen
Glenn Parsons
Ronald C. Petersen
Narayanan Ramachandran
Jon Walter Rosdahl
Sam Sciacca

*Member Emeritus

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

Howard L. Wolfman, *TAB Representative*
Michael Janezic, *NIST Representative*
Satish K. Aggarwal, *NRC Representative*

Lisa Perry
IEEE Standards Program Manager, Document Development

Malia Zaman
IEEE Standards Program Manager, Technical Program Development

Contents

IEEE Standard Classification for Software Anomalies

1. Overview

1.1 Scope

This standard provides for the core set of attributes for classification of failures and defects. It is recognized that there are other attributes of failures or defects that are of unique value to specific applications or business requirements. This standard is applicable to any software (including operating systems, database management systems, applications, testware, firmware, and embedded software) and to any phase of the project, product, or system life cycle by which the software is developed, operated, and sustained. Classification attributes are unaffected by the choice of life cycle model, and that choice is outside the scope of this standard. Some tailoring of classification attribute values based on the chosen life cycle is expected and consistent with the intent of this standard.

1.2 Purpose

The purpose of this standard is to define a common vocabulary with which different people and organizations can communicate effectively about software anomalies and to establish a common set of attributes that support industry techniques for analyzing software defect and failure data.

1.3 Field of application

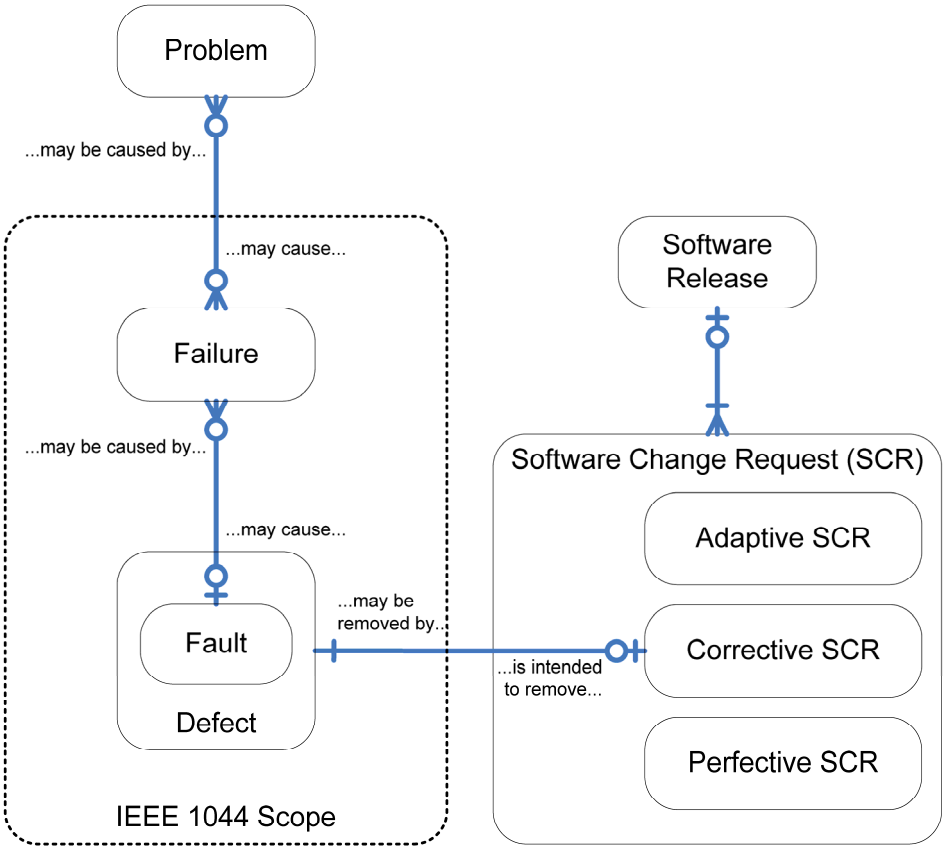
As depicted in Figure 1[1] and Figure 2, problems may be precursors to failure recognition. These are the conditions by which a user recognizes that the software is performing in an undesirable manner. Similarly,

[1] Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

actions taken in response to a failure and fault may be documented as a change request. The classification of problems and change requests is outside of the scope of this standard.

Understanding the scope of this standard depends on understanding the definitions in Clause 2, so it is advisable to read them before proceeding. Scope understanding is also dependent on understanding the relationship among several conceptual entities, which are depicted graphically in Figure 1 and Figure 2 and described textually in Table 1.

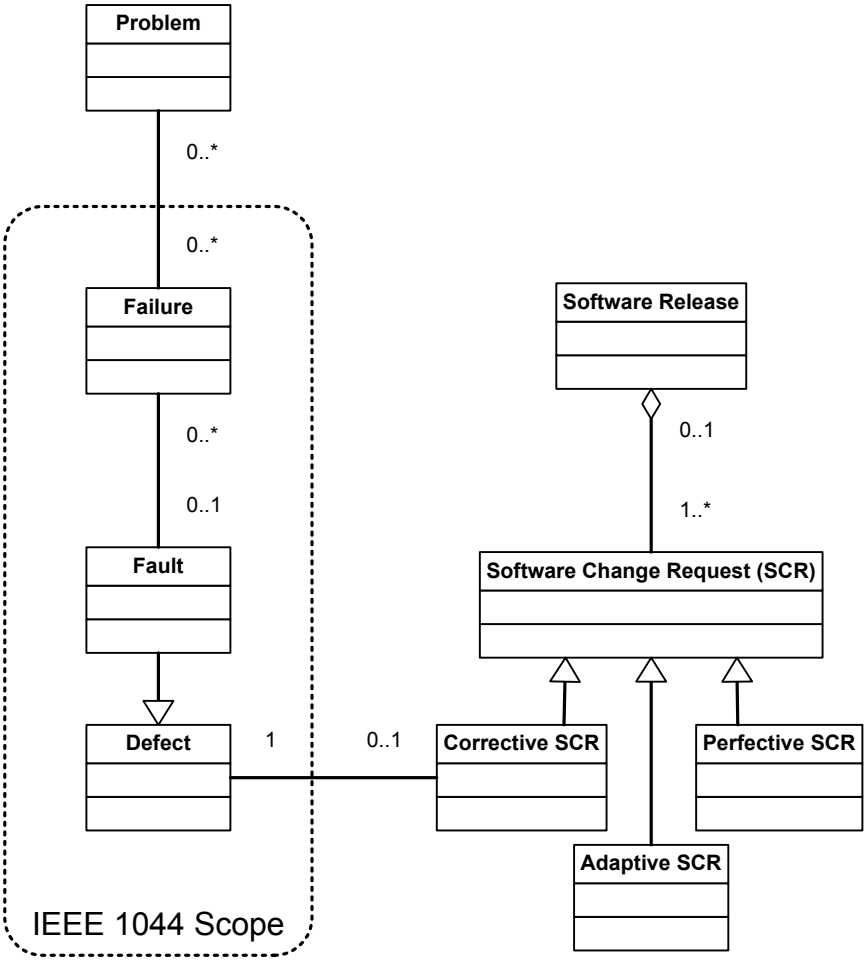
Although software change request (SCR) and software release are not addressed within this standard, they are included in the diagrams to help clarify the scope. As examples, these could have been shown in several different configurations. The one used here is the relatively simple case in which a defect may be associated with a single corrective SCR and each SCR may be associated with, at the most, a single release.



NOTE 1—The rounded rectangles represent entities (things of interest) and the lines connecting the rectangles represent relationships between entities. The symbols at the line ends indicate the number of entities at the end of the line. An open circle near a line end indicates that as few as zero are permitted (i.e., participation is optional); the absence of the circle indicates at least one is required (i.e., participation is mandatory). The three-legged "crows feet" symbol indicates that many entities are permitted to participate; the absence of the crows feet symbol indicates that no more than one entity may participate. A rounded rectangle appearing within another rounded rectangle indicates a parent–child relationship, wherein the contained entity is a subtype of the containing entity (supertype). The relationships represented graphically in this diagram are further described in Table 1.

NOTE 2—This diagram is not intended to mandate a particular notational methodology, and it is not intended as a schema for a database.

Figure 1—Relationships modeled as an entity relationship diagram



NOTE 1—The rectangles represent object classes (things of interest), and the lines connecting the rectangles represent relationships between classes. The three sections within each rectangle contain (from top to bottom) the name, attributes, and methods/operations of the corresponding class. Because the primary focus of this diagram is the relationships, only the class name is included. The methods are outside the scope of this standard, and the attributes are listed and defined in the clauses that follow. The numbers beside the lines indicate the multiplicity of the relationship, with "1" meaning exactly one, "0..1" meaning zero or one, "1..*" meaning one or more, and "0..*" meaning zero, one, or more. Lines with an open triangle on one end indicate a generalization (parent–child) relationship between a supertype class and the subtype class at the other end. Lines with a diamond at the end indicate that more than one change request may be included in one release. The relationships represented graphically in this diagram are further described in Table 1.

NOTE 2—This diagram is not intended to mandate a particular notational methodology, and it is not intended as a schema for a database.

Figure 2—Relationships modeled as a UML class diagram

Table 1—Relationships

Class/entity pair	Relationships
Problem-Failure	A problem may be caused by one or more failures. A failure may cause one or more problems.
Failure-Fault	A failure may be caused by (and thus indicate the presence of) a fault. A fault may cause one or more failures.
Fault-Defect	A fault is a subtype of the supertype defect. Every fault is a defect, but not every defect is a fault. A defect is a fault if it is encountered during software execution (thus causing a failure). A defect is not a fault if it is detected by inspection or static analysis and removed prior to executing the software.
Defect-Change Request	A defect may be removed via completion of a corrective change request. A corrective change request is intended to remove a defect. (A change request may also be initiated to perform adaptive or perfective maintenance.)

The life cycle of a defect is depicted in Figure 3. As defects cannot be classified until they are detected, this standard addresses classification of defects that have been detected and classification of failures that indicate the presence of defects. The methods and processes for detecting and removing defects and for investigating and resolving failures are outside the scope of this standard. Similarly, the process for determining whether a defect should be removed is also outside of the scope for this standard.

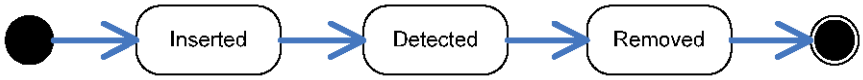


Figure 3—Defect life cycle as a UML statechart diagram

Table 2 summarizes the scope of the standard from another perspective by listing several objectives as being either in scope or out of scope.

Table 2—Scope delineation

In scope	Out of scope
Classifying defects	Classifying corrective actions
Classifying faults	Classifying errors
Classifying failures	Classifying problems
Defining a core set of widely applicable classification attributes	Defining all potentially useful classification attributes
Defining sample attribute values to facilitate understanding	Tailoring sample attribute values to meet specific organizational needs
	Defining when during a project or product life cycle to initiate a formal classification process
	Defining a process that prescribes who should decide what value to assign to which attribute
	Defining a process that prescribes who should record attribute values, when, where, or how
	Disposition process of whether or not to remove the defect

2. Definitions

For the purposes of this document, the following terms and definitions apply. *The IEEE Standards Dictionary: Glossary of Terms and Definitions* should be referenced for terms not defined in this clause.[2]

defect: An imperfection or deficiency in a work product where that work product does not meet its requirements or specifications and needs to be either repaired or replaced. (adapted from the Project Management Institute [B19][3])

NOTE—Examples include such things as 1) omissions and imperfections found during early life cycle phases and 2) faults contained in software sufficiently mature for test or operation.

error: A human action that produces an incorrect result. (adapted from ISO/IEC 24765:2009 [B17])

failure: (A) Termination of the ability of a product to perform a required function or its inability to perform within previously specified limits. (adapted from ISO/IEC 25000:2005 [B18]) **(B)** An event in which a system or system component does not perform a required function within specified limits. (adapted from ISO/IEC 24765:2009 [B17])

NOTE—A failure may be produced when a fault is encountered.

fault: A manifestation of an error in software. (adapted from ISO/IEC 24765:2009 [B17])

problem: (A) Difficulty or uncertainty experienced by one or more persons, resulting from an unsatisfactory encounter with a system in use. **(B)** A negative situation to overcome. (adapted from ISO/IEC 24765:2009 [B17])

3. Classification

3.1 Classification process

The organization shall define its classification process as follows:

a) The goal(s) to be achieved by classifying defects and failures.
b) The reference standard (e.g., as described in a specification, contract, or plan) used to determine which system/software behaviors constitute failure.
c) How disagreement or conflict regarding classification decisions are to be resolved.
d) When classification is to begin and end within the project or product life cycle.
e) The project-, product-, or organization-specific values that are eligible for assignment to classification attributes (see examples in Table A.1 and Table A.2).
f) Who is to assign values to the classification attributes listed in Table 3 and Table 4 for each defect and failure discovered, respectively.
g) Where and how classification data are to be maintained.

3.2 Defect classification

The organization shall record values for all defect attributes listed in Table 3. This set of attributes is not intended to be exhaustive. Sample values for selected attributes are listed in Table A.1. These values are

[2] *The IEEE Standards Dictionary: Glossary of Terms & Definitions* is available at http://shop.ieee.org/.
[3] The numbers in brackets correspond to those of the bibliography in Annex C.

informative only. The values for attributes are likely to be recorded over time as the organization addresses the defect. The status of a defect can be Inserted, Detected, or Removed (see Figure 3); however, it is common to track the status of the associated defect report instead of the status of the defect, so the value assigned to the status typically refers to the state within an organization's workflow related to its defect resolution process. No mandatory status values are prescribed, as there are many organization-specific defect workflows.

Table 3—Defect attributes

Attribute	Definition
Defect ID	Unique identifier for the defect.
Description	Description of what is missing, wrong, or unnecessary.
Status	Current state within defect report life cycle.
Asset	The software asset (product, component, module, etc.) containing the defect.
Artifact	The specific software work product containing the defect.
Version detected	Identification of the software version in which the defect was detected.
Version corrected	Identification of the software version in which the defect was corrected.
Priority	Ranking for processing assigned by the organization responsible for the evaluation, resolution, and closure of the defect relative to other reported defects.
Severity	The highest failure impact that the defect could (or did) cause, as determined by (from the perspective of) the organization responsible for software engineering.
Probability	Probability of recurring failure caused by this defect.
Effect	The class of requirement that is impacted by a failure caused by a defect.
Type	A categorization based on the class of code within which the defect is found or the work product within which the defect is found.
Mode	A categorization based on whether the defect is due to incorrect implementation or representation, the addition of something that is not needed, or an omission.
Insertion activity	The activity during which the defect was injected/inserted (i.e., during which the artifact containing the defect originated).
Detection activity	The activity during which the defect was detected (i.e., inspection or testing).
Failure reference(s)	Identifier of the failure(s) caused by the defect.
Change reference	Identifier of the corrective change request initiated to correct the defect.
Disposition	Final disposition of defect report upon closure.

3.3 Failure classification

The organization shall record values for all failure attributes listed in Table 4. The sample values listed in Table A.2 are informative only.

Table 4—Failure attributes

Attribute	Definition
Failure ID	Unique identifier for the failure.
Status	Current state within failure report life cycle. See Table B.1.
Title	Brief description of the failure for summary reporting purposes.
Description	Full description of the anomalous behavior and the conditions under which it occurred, including the sequence of events and/or user actions that preceded the failure.
Environment	Identification of the operating environment in which the failure was observed.
Configuration	Configuration details including relevant product and version identifiers.
Severity	As determined by (from the perspective of) the organization responsible for software engineering. See Table B.1.
Analysis	Final results of causal analysis on conclusion of failure investigation.
Disposition	Final disposition of the failure report. See Table B.1.
Observed by	Person who observed the failure (and from whom additional detail can be obtained).
Opened by	Person who opened (submitted) the failure report.
Assigned to	Person or organization assigned to investigate the cause of the failure.
Closed by	Person who closed the failure report.
Date observed	Date/time the failure was observed.
Date opened	Date/time the failure report is opened (submitted).
Date closed	Date/time the failure report is closed and the final disposition is assigned.
Test reference	Identification of the specific test being conducted (if any) when the failure occurred.
Incident reference	Identification of the associated incident if the failure report was precipitated by a service desk or help desk call/contact.
Defect reference	Identification of the defect asserted to be the cause of the failure.
Failure reference	Identification of a related failure report.

Annex A

(informative)

Example values for attributes

Table A.1 —Examples of defect attribute values

Attribute	Value	Definition
Status	Open	Future action is anticipated in response to a detected defect.
Status	Closed	No further action is planned (regardless of whether the defect has been removed).
Priority	High	Defects with this rating received top priority for analysis and resolution.
Priority	Medium	Defects with this rating are in the queue for analysis and resolution behind those with a high-priority rating.
Priority	Low	Defects with this rating are at the end of the queue for analysis and resolution.
Severity	Blocking	Testing is inhibited or suspended pending correction or identification of suitable workaround.
Severity	Critical	Essential operations are unavoidably disrupted, safety is jeopardized, and security is compromised.
Severity	Major	Essential operations are affected but can proceed.
Severity	Minor	Nonessential operations are disrupted.
Severity	Inconsequential	No significant impact on operations.
Probability	High	A likelihood of occurrence greater than 70%.
Probability	Medium	A likelihood of occurrence between 40% and 70%.
Probability	Low	A likelihood of occurrence less than 40%.
Effect	Functionality	Actual or potential cause of failure to correctly perform a required function (or implementation of a function that is not required), including any defect affecting data integrity.
Effect	Usability	Actual or potential cause of failure to meet usability (ease of use) requirements.
Effect	Security	Actual or potential cause of failure to meet security requirements, such as those for authentication, authorization, privacy/confidentiality, accountability (e.g., audit trail or event logging), and so on.
Effect	Performance	Actual or potential cause of failure to meet performance requirements (e.g., capacity, computational accuracy, response time, throughput, or availability).
Effect	Serviceability	Actual or potential cause of failure to meet requirements for reliability, maintainability, or supportability (e.g., complex design, undocumented code, ambiguous or incomplete error logging, etc.).
Effect	Other	Would/does not cause any of the above effects.
Type	Data	Defect in data definition, initialization, mapping, access, or use, as found in a model, specification, or implementation. Examples: Variable not assigned initial value or flag not set Incorrect data type or column size Incorrect variable name used Valid range undefined Incorrect relationship cardinality in data model Missing or incorrect value in pick list

Attribute	Value	Definition
Type	Interface	Defect in specification or implementation of an interface (e.g., between user and machine, between two internal software modules, between software module and database, between internal and external software components, between software and hardware, etc.). Examples: Incorrect module interface design or implementation Incorrect report layout (design or implementation) Incorrect or insufficient parameters passed Cryptic or unfamiliar label or message in user interface Incomplete or incorrect message sent or displayed Missing required field on data entry screen
Type	Logic	Defect in decision logic, branching, sequencing, or computational algorithm, as found in natural language specifications or in implementation language. Examples: Missing else clause Incorrect sequencing of operations Incorrect operator or operand in expression Missing logic to test for or respond to an error condition (e.g., return code, end of file, null value, etc.) Input value not compared with valid range Missing system response in sequence diagram Ambiguous definition of business rule in specification
Type	Description	Defect in description of software or its use, installation, or operation.
Type	Syntax	Nonconformity with the defined rules of a language.
Type	Standards	Nonconformity with a defined standard.
Type	Other	Defect for which there is no defined type.
Mode	Wrong	Something is incorrect, inconsistent, or ambiguous.
Mode	Missing	Something is absent that should be present.
Mode	Extra	Something is present that need not be.
Insertion activity	Requirements	Defect inserted during requirements definition activities (e.g., elicitation, analysis, or specification). Examples: Function required to meet customer goals omitted from requirements specification Incomplete use case specification Performance requirements missing or incorrect Security requirements missing or incorrect Function incorrectly specified in requirements specification Function not needed to meet customer goals specified in requirements specification
Insertion activity	Design	Defect inserted during design activities. Examples: Design incapable of supporting stated requirements Incorrect derivation of physical data model from logical data model Incorrect application program interface design
Insertion activity	Coding	Defect inserted during "coding" or analogous activities. Examples: Incorrect variable typing Incorrect data initialization Module interface not coded as designed

Attribute	Value	Definition
Insertion activity	Configuration	Defect inserted during product build or packaging. Examples: Wrong source file included in build Wrong .EXE file included in distribution/deployment package Wrong localization parameters in .INI file
Insertion activity	Documentation	Defect inserted during documentation of instructions for installation or operation. Examples: Incorrect menu choices listed in User Manual Incorrect task or navigation instructions in on-line help Missing installation pre-requisite in product specifications Wrong version identifier in product release notes
Detection activity	Requirements	Defect detected during synthesis, inspection, or review of requirements.
Detection activity	Design	Defect detected during synthesis, inspection, or review of design.
Detection activity	Coding	Defect detected during synthesis, inspection, or review of source code.
Detection activity	Supplier testing	Defect detected during any testing conducted by the supplier.
Detection activity	Customer testing	Defect detected during any testing conducted by the customer.
Detection activity	Production	Defect detected during production operation and use
Detection activity	Audit	Defect detected during an audit (prerelease or postrelease).
Detection activity	Other	Defect detected during any other activity, such as user/operator training or product demonstrations.
Disposition	Corrected	Defect was corrected/removed.
Disposition	Not found	No defect was found. Failure could not be reproduced, or the reported behavior is actually intended behavior.
Disposition	Referred	Defect is contained in another organization's asset and was referred to that organization for correction.
Disposition	Duplicate	Defect report is a duplicate.

Table A.2 —Examples of failure attributes values

Attribute	Value	Definition
Status	Open	Future action is anticipated.
Status	Closed	No further action is planned.
Severity	Critical	Essential operations are unavoidably disrupted and/or safety is jeopardized.
Severity	Major	Essential operations are affected but can proceed.
Severity	Minor	Nonessential operations are disrupted.
Severity	Inconsequential	No significant impact on operations.
Disposition	Cause unknown	No failure cause found; failure symptom(s) ceased.
Disposition	Duplicate	Another report of the same failure event already exists.
Disposition	Resolved	Failure cause found and resolved.

Annex B

(informative)

Classification examples

Example problems are listed as follows, and their associated classification data are listed in Table B.1.

Problem 1: Sue calls service desk and reports she cannot log in to timesheet system because the password field is missing from the login screen. *{In this example, Sue has a problem in that she cannot log in, caused by a failure wherein the password field did not appear on the login screen, which was in turn caused by a defect inserted during coding of the Login.asp artifact.}*

Problem 2: Joe calls service desk and reports he cannot log in to timesheet system because the password field is missing from the login screen. *{This example is similar to Problem 1 and serves to illustrate that two distinct failures (events) can be caused by a single defect (condition).}*

Problem 3: During customer qualification testing, Sam observed that the color of the font does not match the requirements document section 4.2.1.3. *{This example illustrates the difference between the failure (appearance of incorrect color on screen) and the defect that caused it (incorrect data value assigned to a constant in the code).}*

Problem 4: During a peer review for software requirements for a new financial management system, Alice discovers that values are in the requirements as thousands of dollars instead of as millions of dollars. *{This example illustrates classification of a defect detected directly, prior to any failure occurring.}*

Problem 5: Company A's battery ran out of power because there was no low-power warning. The design of a security system monitoring system did not include a warning for low battery power, despite the fact that this feature was specified in the requirements. *{In this example, the defect was not detected until a failure occurred in a production environment.}*

In Table B.1, the column numbers correspond to the numbered example problems, and the cell at the intersection of the classification attribute (row) and problem (column) contains the example attribute value.

Table B.1 —Classification examples

Entity	Attribute	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5
Failure	Failure ID	F080001	F080002	FID001	N/A	DID005
Failure	Status	Open	Open	Open	N/A	Closed
Failure	Title	Failure: missing password field	Failure: missing password field—duplicate	Display—wrong color font	N/A	Missing low battery power alert
Failure	Description	The password field is missing from the login screen	The password field is missing from the login screen	Font color does not meet system specifications: black instead of blue	N/A	A low battery power alert is missing from Company A's security monitoring system
Failure	Environment	Chicago-websrvr23	Chicago-websrvr23	Windows XP	N/A	A_sysmon_001
Failure	Configuration	TimeSheet v6.4	TimeSheet v6.4	Display_version_4.0	N/A	Power_config_v01
Failure	Severity	Critical	Critical	Minor	N/A	Critical
Failure	Analysis	Code error		Color of font does not match requirement specification section 4.2.1.3	N/A	Design did not include low battery power
Failure	Disposition		Duplicate		N/A	
Failure	Observed by	Sue	Joe	Sam	N/A	Thomas
Failure	Opened by	Williams	Hartley	Jones	N/A	Sam
Failure	Assigned to			Smith	N/A	Joe
Failure	Closed by					Marsha
Failure	Date observed	April 1, 2008	April 2, 2008	October 10, 2007	N/A	Feb 4, 2006
Failure	Date opened	April 1, 2008	April 2, 2008	October 12, 2007	N/A	Feb 5, 2006
Failure	Date closed				N/A	June 9, 2006
Failure	Test reference	N/A	N/A	Disp_font_ver_1	N/A	Securemon 09
Failure	Incident reference	S080002	S080003	HD001	N/A	SEC054
Failure	Defect reference	D080234	D080234	F080001	N/A	SD089
Defect	Defect ID	D080234	N/A	C080049	FM003	SD089
Defect	Description	Password field not correctly implemented in code	N/A	Constant containing hexadecimal http color code was 000000 instead of 0000FF	Incorrect monetary units specified in requirements	Design did not include low-battery alert according to Company A's requirements
Defect	Status	Open	N/A	Open	Closed	Closed

Entity	Attribute	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5
Defect	Asset	TS-srvr	N/A	StockTradeR	FinanceForAll	SecureIT
Defect	Artifact	Login.asp	N/A	Cdisplay.c	FMSYS_Reqs	Design Specification
Defect	Version detected	V6.4	N/A	V3.4	Initial	V1.0
Defect	Version corrected		N/A	V4.0	V1.0	V2.3
Defect	Priority			Low	High	High
Defect	Severity	Critical	N/A	Minor	Major	Major
Defect	Probability	High	N/A	High	Low	High
Defect	Effect	Functionality	N/A	Usability	Functionality	Functionality
Defect	Type	Interface	N/A	Data	Data	Other
Defect	Mode	Missing	N/A	Wrong	Wrong	Missing
Defect	Insertion activity	Coding	N/A	Coding	Requirements	Design
Defect	Detection activity	Production	N/A	Customer Testing	Peer review	Production
Defect	Failure reference	F080001	N/A	FID001	N/A	DID005
Defect	Change reference	C080049	N/A	C_052	CHG_005	SWC_005
Defect	Disposition				Corrected	Corrected

Annex C

(informative)

Bibliography

[B1] Florac, W. A., *Software Quality Measurement: A Framework for Counting Problems and Defects* (CMU/SEI-92-TR-22). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1992.

[B2] Grady, R. B., and Caswell, D. L., *Software Metrics: Establishing a Company-Wide Program*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[B3] Grady, R. B., *Practical Software Metrics for Project Management and Process Improvement*. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[B4] Humphrey, W. S., *Managing the Software Process*. Reading, MA: Addison-Wesley, 1989.

[B5] IEEE Std 730™-2002, IEEE Standard for Software Quality Assurance Plans.[4, 5]

[B6] IEEE Std 828™-2005, IEEE Standard for Software Configuration Management Plans.

[B7] IEEE Std 830™-1998, IEEE Recommended Practice for Software Requirements Specifications.

[B8] IEEE Std 1008™-1987, IEEE Standard for Software Unit Testing.

[B9] IEEE Std 1012™-2004, IEEE Standard for Software Verification and Validation.

[B10] IEEE Std 1028™-1997, IEEE Standard for Software Reviews.

[B11] IEEE Std 1061™-1998, IEEE Standard for a Software Quality Metrics Methodology.

[B12] IEEE Std 1074™-2006, IEEE Standard for Developing a Software Project Life Cycle Process.

[B13] ISO/IEC 12207-2008, Systems and Software Engineering—Software Life Cycle Processes.[6]

[B14] ISO/IEC 14143-1-1998, Information Technology—Software Measurement–Functional Size Measurement—Part 1: Definition of Concepts.

[B15] ISO/IEC 15288-2008, Systems and Software Engineering—System Life Cycle Processes.

[B16] ISO/IEC 15939-2007, Systems and Software Engineering—Measurement Process.

[B17] ISO/IEC 24765-2009, Systems and Software Engineering—Vocabulary.

[4] The IEEE standards or products referred to in this annex are trademarks owned by the Institute of Electrical and Electronics Engineers, Incorporated.
[5] IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org/).
[6] ISO publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Geneva 20, Switzerland (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/).

[B18] ISO/IEC 25000-2005, Software Engineering–Software Product Quality Requirements and Evaluation (SQuaRE)—Guide to SQuaRE.

[B19] Project Management Institute, *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 4th ed. Newtown Square, PA: Project Management Institute, 2008.