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)

This is a Redline Document produced by Thomson Reuters, Techstreet.

This document is intended to provide users with an indication of changes from one edition to the next. It includes a full-text version of the new document, plus an indication of changes from the previous version.

Redlines are designed to save time and improve efficiencies by using the latest software technology to find and highlight document changes. More professionals are using valuable new technologies like redlines, to help improve outcomes in a fastpaced global business world.

Because it may not be technically possible to capture all changes accurately, it is recommended that users consult previous editions as appropriate. In all cases, only the current base version of this publication is to be considered the official document.

Redline Processing Notes:

1. ~~**Red Text**~~ **-** Red strikethrough text denotes deletions.

2. **Blue Text** **-** Blue underlined text denotes modifications and additions.

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.

~~This standard is applicable to any software, including critical computer software, commercial applications, system software, support software, testware, and firmware during any phase of a system's life cycle. This standard defines the minimum requirements for classifying anomalies, as well as providing additional classifications for projects requiring greater detail. The mandatory classifications are the minimum requirements necessary to establish a standard terminology for anomalies within or between projects and organizations~~

~~To accomplish the classification task, this standard documents the following subjects:~~

~~a) Definitions of terms not provided in IEEE Std 610.12-1990[1]~~
~~b) A basic process (sequence of steps) for classifying and establishing categories of anomalies relating to software products and providing related data and information~~
~~c) A standard set of categories and classifications, either mandatory or optional~~
~~d) A list of supporting data items~~

~~1.1 Background~~

~~This standard is based on several definitions, concepts, and processes that need to be understood prior to its use. These are discussed in the following paragraphs. Formal definitions can be found in clause 3.~~

~~A key term in this standard is anomaly. An anomaly is any condition that departs from the expected. This expectation can come from documentation (requirements specifications, design documents, user documents, standards, etc.) or from someone's perceptions or experiences. An anomaly is not necessarily a problem in the software product; it may be manifesting correct behavior in which case changing the software would be an enhancement. An anomaly may also be caused by something other than the software. For reasons of semantics, use of the word *anomaly* is preferred over the words *error, fault, failure, incident, flaw, problem, gripe, glitch, defect,* or *bug* throughout this standard because it conveys a more neutral connotation.~~

~~The methodology of this standard is based on a process (sequence of steps) that pursues a logical progression from the initial recognition of an anomaly to its final disposition. Each step interrelates with and sup- ports the other steps. This process is graphically displayed in figure 1, and a complete discussion of the classification process is in 4.1. This standard does not attempt to enforce any particular anomaly processing procedure other than to identify the basic processes an anomaly goes through. It is expected that users will modify the process based on their organization's procedures. Subclause 4.2 describes the classification scheme and supporting data items useful for identifying, tracking, and resolving anomalies.~~

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

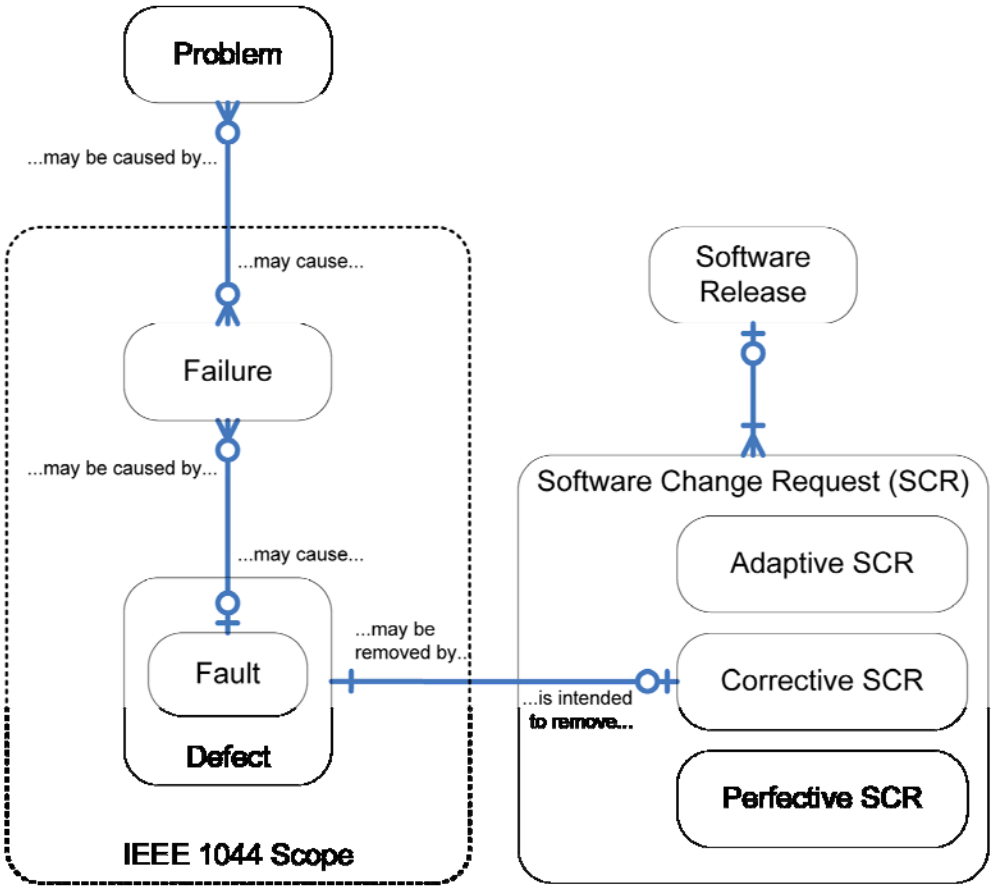
As depicted in Figure 1[1] and Figure 2, problems may be precursors to failure recognition. These are the conditions by which a user recognizes that the software is performing in an undesirable manner. Similarly,

[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.

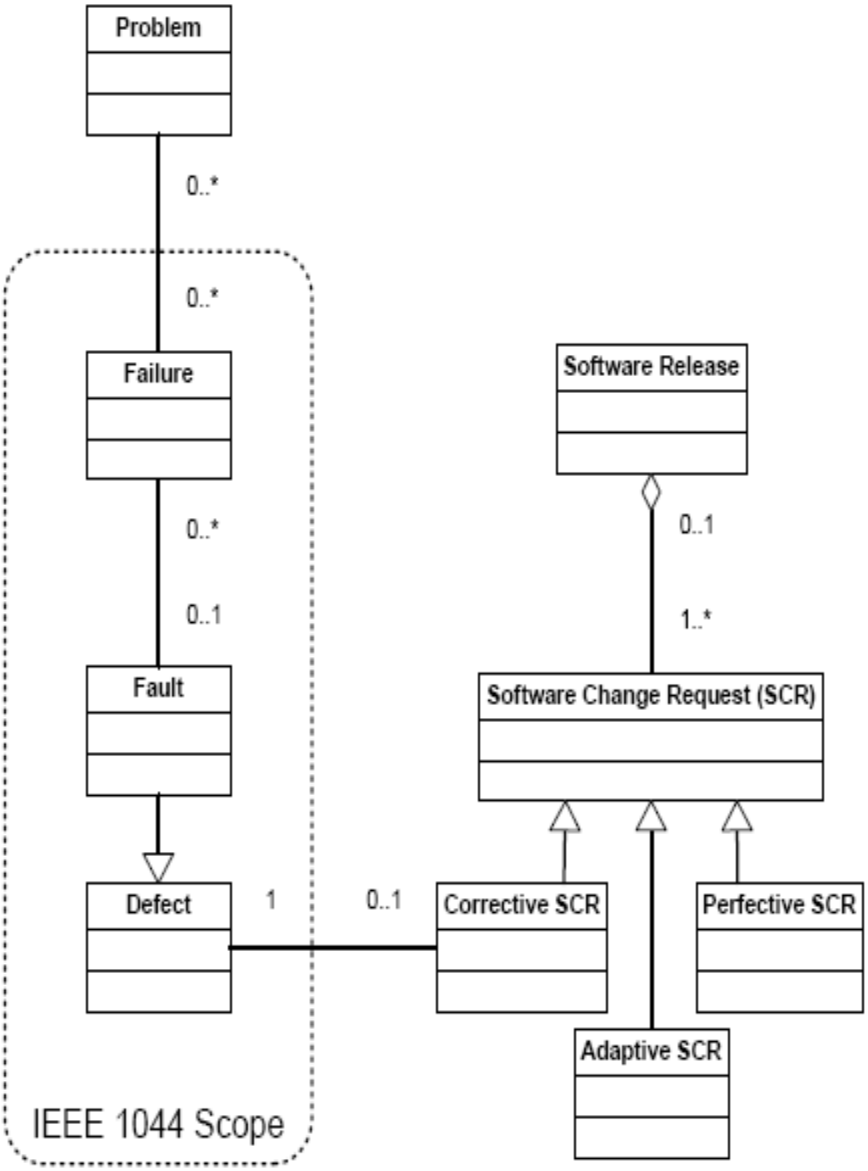
Although software change request (SCR) and software release are not addressed within this standard, they are included in the diagrams to help clarify the scope. As examples, these could have been shown in several different configurations. The one used here is the relatively simple case in which a defect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

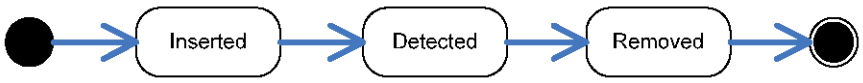


Figure 3—Defect life cycle as a UML statechart diagram

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])

~~Definitions of terms used in this standard are consistent with IEEE Std 610.12-1990, with the few exceptions noted below. Note that the term *anomaly* as defined in this standard expands upon the limited definition included in IEEE Std 610.12-1990. This standard's definition of anomaly includes deviations from the user's perceptions or experiences.~~

~~**3.1 anomaly:** Any condition that deviates from expectations based on requirements specifications, design documents, user documents, standards, etc. or from someone's perceptions or experiences. Anomalies may be found during, but not limited to, the review, test, analysis, compilation, or use of software products or applicable documentation. belongs~~

~~**3.2 category:** An attribute of an anomaly to which a group of classifications~~

~~**3.3 classification:** A choice within a category.~~

~~**3.4 classification process:** The classification process is a series of activities, starting with the recognition of an anomaly through to its closure. The process is divided into four sequential steps interspersed with three administrative activities. The sequential steps are as follows:~~

~~a) Step 1: Recognition~~
~~b) Step 2: Investigation~~
~~c) Step 3: Action~~
~~d) Step 4: Disposition~~

The three administrative activities applied to each sequential step are as follows:

a) Recording
b) Classifying
c) Identifying impact

3.5 mandatory category: A category that is essential to establish a common definition and to provide common terminology and concepts for communication among projects, business environments, and personnel.

3.6 optional category: A category that provides additional details that are not essential but may be useful in particular situations.

3.7 supporting data item: Data used to describe an anomaly and the environment in which it was encountered.

2. References

This standard shall be used with the following publications:

IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology (ANSI).

Brehmer, C. L., Carl, J. R., "Incorporating IEEE Std 1044 into Your Anomaly Tracking Process," *Crosstalk*, no. 40, pp. 9-16, Jan. 1993.

[1]Information on references can be found in clause 2.

3. Classification

3.1 Classification process

~~This standard refers to the anomaly classification process shown in figure 1. The standard classification scheme is described in 4.2. The standard categories and classifications are shown in the odd-numbered tables. The supporting data items are shown in the even-numbered tables.~~

~~Because the classification scheme depends on an understanding of the basic classification process, the classification process will be discussed first.~~

The organization shall define its classification process ~~is a series of activities, starting with the recognition of an anomaly through to its closure. The process is divided into four sequential steps interspersed with three administrative activities. The sequential steps are~~ 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.

~~a) Step 1: Recognition~~
~~b) Step 2: Investigation~~
~~c) Step 3: Action~~
~~d) Step 4: Disposition~~

~~The three administrative activities applied to each sequential step are as follows:~~

~~a) Recording~~
~~b) Classifying~~
~~c) Identifying impact~~

~~Each of the steps and administrative activities is discussed below. Figure 1 illustrates the process and interrelationships between the activities. It is anticipated that in the processing of any one anomaly these steps may be repeated several times. It is also expected that individual organizations implementing this standard will use procedures that accomplish the standard's objective but may call for additional detail, a different sequence of steps, and/or may specify organizational involvement~~

4.1.1 Recognition

The recognition step occurs when an anomaly is found. Recognition of an anomaly may be made by anyone involved in the development, evaluation, use, support, or maintenance of software, regardless of where in the software life cycle the anomaly is discovered.

a) *Recording the recognition.* When an anomaly is recognized, supporting data items are recorded to identify the anomaly and the environment in which it occurred. The applicable supporting data items indicated in table 2 are recorded as part of the recognition process.
b) *Classifying the recognition.* Certain important attributes of an anomaly are observed and shall be identified during the recognition process. Tables 1a–1f contain the categories of information and the scheme by which these attributes shall be classified.
c) *Identifying impact.* The person discovering the anomaly shall identify his or her perception of the impact of the anomaly. Tables 7a–7i list the categories used for impact identification.

4.1.2 Investigation

Following the recognition step, each anomaly shall be investigated. This investigation shall be of sufficient depth either to identify all known related issues and propose solutions or to indicate that the anomaly requires no action.

a) *Recording the investigation.* The supporting data items indicated as in table 4 are recorded as part of the investigation process. Any new information related to the recognition of the anomaly is updated.
b) *Classifying the investigation.* During the investigation, entries shall be provided for all categories identified as mandatory in tables 3a–3c. In addition, classification entries made during the recognition step shall be reviewed and corrected where appropriate.
c) *Identifying impact.* Previous impact classifications shall be reviewed and updated based on the results of the analysis. Tables 7a–7i list the categories used for impact identification.

4.1.3 Action

A plan of action shall be established based on the results of the investigation. The action includes all activities necessary to resolve the immediate anomaly and those activities required to revise processes, policies, or other conditions necessary to prevent the occurrence of similar anomalies.

a) *Recording the action.* The supporting data items necessary to record the decisions on how to handle the anomaly are in table 6. The applicable supporting data items in this section are recorded.
b) *Classifying the action.* Once the appropriate action or actions are determined, the mandatory categories in tables 5a and 5b shall be classified. Note that the action taken to resolve an anomaly is related to the disposition of the anomaly.
c) *Identifying impact.* Classifications chosen for categories determined in previous steps shall be reviewed and updated as required. Tables 7a–7i list the categories used for impact identification.

4.1.4 Disposition

Following the completion of either all required resolution actions or at least identification of long-term corrective action, each anomaly shall be disposed of by:

a) *Recording the disposition.* The applicable supporting data items in table 10 are recorded for each anomaly.
b) *Classifying the disposition.* Anomalies shall be completed using one of the disposition classifications shown in table 9.
c) *Identifying impact.* Previously recorded impact categories shall be reviewed and updated. All classifications shown in tables 7a–7i as mandatory shall be completed.

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.

~~4.2 Standard classification scheme~~

~~The following subclauses describe how the classification scheme is used in the odd numbered tables. The classification scheme is structured around the process steps described in figure 1, including recognition, investigation, action, recording, classifying, impact identification, and disposition.~~

~~Throughout the classification scheme, there are some occasions when a reference is required. Also, there are instances where more than one classification may be applicable within a category, such as *Type*. In these cases, more than one classification can be indicated, though only one classification per category is normally necessary.~~

4.2.1 Classification codes

The classification codes, primarily listed in the third column of the odd numbered tables, are an alphanumeric code (useful for data entry purposes) for each of the categories and classifications shown in the first and fourth columns of the tables, respectively.

Each of the steps or activities being classified is assigned a two character alpha prefix: RR for Recognition step, IV for Investigation step, AC for Action step, IM for Impact Identification activity, and DP for Disposition step.

The prefix is followed by three digits, identifying the categories and classifications. Where further definition is needed, a decimal number is assigned. For instance, in table 3c classification code IV321.1 first guides the user to the Investigation step (IV). Second, the category the classification belongs to is *Type* (IV300). The *Type* of anomaly is identified as a *Computational problem* (IV320), further identified as an *Equation insuffi cient or incorrect* (IV321), and, more specifically, as a *Missing computation* (IV321.1).

The supporting data items are necessary for recognition, investigation, action, and disposition of an anomaly. Since each project offers a unique set of relevant supporting data items, this standard does not mandate specific supporting data items. Suggested supporting data items pertinent to each process step are in the even-numbered tables.

4.2.2 Compliance required

In the odd numbered tables is a column marked *Compliance required*. This identifies the importance (mandatory or optional) of the category for compliance to this standard. It is assumed that all applicable classifications within a mandatory category shall be used for compliance to this standard. Only the first level of classifications is mandatory for each mandatory category. Additional classifications may be added when implementing this standard to provide the analyst with needed accuracy.

4.2.3 Recognition

During the recognition process members of a project shall identify the *Project activity* taking place at the time the anomaly was encountered. The *Project activity* category, which is mandatory, includes identifiable activities that are employed during the development and release of a software product. These activities shall be performed during more than one of the project phases. The *Project phase* category is also mandatory and shall be identified and tailored to the project life cycle being employed when implementing this standard. At the current time a consensus has not developed as to which project phases compose the software life cycle. This standard provides software life cycle phase classifications that can be mapped to many specific soft ware life cycles, including the one in IEEE Std 610.12 1990. The *Suspected cause* category provides the anomaly discoverer the opportunity to speculate on the cause of the anomaly. This category is optional. The *Repeatability* category of the anomaly is optional. Classifying the *Symptom* of the anomaly is mandatory. The *Product status* identifies the usability of the product with the anomaly; it is optional.

Recognition categories and classification codes all have an RR prefix, and are shown in tables 1a-1f.

Table 1a-Recognition classification scheme-Project activity

Category	Compliance required	Code	Classification
Project activity RR100	*Mandatory*	RR110	Analysis
	Mandatory	RR120	Review
	Mandatory	RR130	Audit
	Mandatory	RR140	Inspection
	Mandatory	RR150	Code/Compile/Assemble
	Mandatory	RR160	Testing
	Mandatory	RR170	Validation/Qualification testing
	Mandatory	RR180	Support/Operational
	Mandatory	RR190	Walk-through

Table 1b-Recognition classification scheme-Project phase

Category	Compliance required	Code	Classification
Project phase RR200	*Mandatory*	RR210	Requirements
		RR211	Concept evaluation
		RR212	System requirements
		RR213	Software requirements
		RR214	Prototype requirements
	Mandatory	RR220	Design
		RR221	System design
		RR222	Preliminary design
		RR223	Detail design
		RR224	Prototype design
	Mandatory	RR230	Implementation
		RR231	Code
		RR232	Unit test
		RR233	Integrate
		RR234	Prototype
	Mandatory	RR240	Test
		RR241	Integration test
		RR242	System test
		RR243	Beta test
		RR244	Prototype test
		RR245	Acceptance test
		RR246	Installation and checkout
	Mandatory	RR250	Operation and maintenance
	Mandatory	RR260	Retirement

Table 1c-Recognition classification scheme-Suspected cause

Category	Compliance required	Code	Classification
Suspected cause RR300	*Optional*	RR310	Product
		RR311	Hardware
		RR312	Software
		RR313	Data
		RR314	Interface
		RR315	Documentation
		RR316	Enhancement (Perceived inadequacies)
		RR320	Test system
		RR321	Hardware
		RR322	Software
		RR323	Data
		RR324	Interface
		RR325	Documentation
		RR326	Enhancement (Perceived inadequacies)
		RR330	Platform
		RR331	Hardware
		RR332	Operating system
		RR333	Documentation
		RR340	Outside vendor/Third party
		RR341	Hardware
		RR342	Software
		RR343	Data
		RR344	Documentation
		RR345	Enhancement (Perceived inadequacies)
		RR350	User
		RR360	Unknown

Table 1d-Recognition classification scheme-Repeatability

Category	Compliance required	Code	Classification
Repeatability RR400	*Optional*	RR410	One-time occurrence
		RR420	Intermittent
		RR430	Recurring
		RR440	Reproducible
		RR450	Unknown

Table 1e-Recognition classification scheme-System

Category	Compliance required	Code	Classification
Symptom RR500	*Mandatory*	RR510	Operating system crash
	Mandatory	RR520	Program hang-up
	Mandatory	RR530	Program crash
	Mandatory	RR540	Input problem
		RR541	Correct input not accepted
		RR542	Wrong input accepted
		RR543	Description incorrect or missing
		RR544	Parameters incomplete or missing
	Mandatory	RR550	Output problem
		RR551	Wrong format
		RR552	Incorrect result/data
		RR553	Incomplete/Missing
		RR554	Spelling/Grammar
		RR555	Cosmetic
	Mandatory	RR560	Failed required performance
	Mandatory	RR570	Perceived total product failure
	Mandatory	RR580	System error message
	Mandatory	RR590	Other

Table 1f-Recognition classification scheme-Product status

Category	Compliance required	Code	Classification
Product status RR600	*Optional*	RR610	Unusable
		RR620	Degraded
		RR630	Affected, use workaround
		RR640	Unaffected

4.2.3.1 Recognition supporting data items

It is generally necessary to collect considerable information in addition to classifiable information during the recognition process of an anomaly. Table 2 contains a list of important, but optional, supporting data items. Note that this list is not exhaustive.

Table 2-Recognition supporting data items

Environment		Originator	Anomaly definition	Time	Vendor
Product hardware	Name	Name	Anomaly description	Operate time	Company
	Revision	Date anomaly occurred	Customer irritability factor	Wall clock time	Contact
	ID number/model	Code or functional area	Triggering event	System time	Vendor's ID number
	Serial number	E-mail address	Work-around or patch (means to avoid)	CPU time	Expected resolution
Product software	Name	Address	Proposed fix		Expected resolution date
	Version/revision level	Phone number	Documentation used (attached as applicable)		
Database	Name	Company identification			
	Version/revision level	Distribution (activities ID #)			
Test support software	Name				
	Version/revision level				
Platform	Name				
	ID number/model				
	Serial number				
	Version/revision level				
	Operating system version				
Firmware	Name				
	ID number/model				
	Serial number				
	Version/revision level				
Other	Monitor				
	Network				
	Peripherals				

4.2.4 Investigation

The next process step is investigation of the anomaly. The *Actual cause*, *Source*, and the *Type* of the anomaly shall be identified. For instance, if an anomaly report were written because the testing procedure was incorrect, then the *Actual cause* would be IV125 *Test system documentation* and the *Source* would be IV252 *Test procedures*.

The *Type* of anomaly identifies whether the anomaly is in the software documentation, an enhancement, or some aspect of the code itself, such as its logic, interfaces, data handling, data, or computations. There are several levels of the classifications in the *Type* category so that the degree or depth of classification may be chosen for each project beyond the mandatory first level.

Extending the testing procedure incorrect example mentioned in the first paragraph of this subclause, the *Type* could be either IV363, incorrect item in document, or IV316, missing condition test in logic. If test procedures are mostly written in text as opposed to code, the document problem assignation is more appropriate. If test procedures are code or pseudocode, the logic problem classification is better.

Investigation categories and classification codes have an IV prefix, and are listed in tables 3a–3c.

Table 3a–Investigation classification scheme–Actual cause

Category	Compliance required	Code	Classification
Actual cause *IV100*	*Mandatory*	IV110	Product
		IV111	Hardware
		IV112	Software
		IV113	Data
		IV114	Interface
		IV115	Documentation
		IV116	Enhancement (perceived inadequacies)
	Mandatory	IV120	Test system
		IV121	Hardware
		IV122	Software
		IV123	Data
		IV124	Interface
		IV125	Documentation
		IV126	Enhancement (perceived inadequacies)
	Mandatory	IV130	Platform
		IV131	Hardware
		IV132	Operating system
		IV133	Documentation
	Mandatory	IV140	Outside vendor/Third party
		IV141	Hardware
		IV142	Software
		IV143	Data
		IV144	Documentation
		IV145	Enhancement (perceived inadequacies)
	Mandatory	IV150	User
	Mandatory	IV160	Unknown

Table 3b-Investigation classification scheme-Source

Category	Compliance required	Code	Classification
Source IV200	*Mandatory*	IV210	Specification
		IV211	Requirements
		IV212	Functional
		IV213	Preliminary design
		IV214	Detailed design
		IV215	Product design
		IV216	Interface
		IV217	Data
		IV218	Implementation
	Mandatory	IV220	Code
	Mandatory	IV230	Database
	Mandatory	IV240	Manual and guides
		IV241	User guide
		IV242	Reference manual
		IV243	Product internals training manual
		IV244	System administrator manual
		IV245	Installation guide
	Mandatory	IV250	Plans and procedures
		IV251	Test plan
		IV252	Test procedures
		IV253	Quality assurance plan
		IV254	Configuration management plan
		IV255	Maintenance plan
		IV256	Product support plan
	Mandatory	IV260	Reports
		IV261	Test report
		IV262	Quality assessment report
	Mandatory	IV270	Standards/Policies

Table 3c-Investigation classification scheme-Type

Category	Compliance required	Code	Classification
Type IV300	*Mandatory*	IV310	Logic problem
		IV311	Forgotten cases or steps
		IV312	Duplicate logic
		IV313	Extreme conditions neglected
		IV314	Unnecessary function
		IV315	Misinterpretation
		IV316	Missing condition test
		IV317	Checking wrong variable
		IV318	Iterating loop incorrectly
	Mandatory	IV320	Computation problem
		IV321	Equation insufficient or incorrect
		IV321.1	Missing computation
		IV321.2	Operand in equation incorrect
		IV321.3	Operator in equation incorrect
		IV321.4	Parentheses used incorrectly
		IV322	Precision loss
		IV322.1	Rounding or truncation fault
		IV322.2	Mixed modes
		IV323	Sign convention fault
	Mandatory	IV330	Interface/timing problem
		IV331	Interrupts handled incorrectly
		IV332	I/O timing incorrect
		IV332.1	Timing fault causes data loss
		IV333	Subroutine/Module mismatch
		IV333.1	Wrong subroutine called
		IV333.2	Incorrectly located subroutine call
		IV333.3	Nonexistent subroutine called
		IV333.4	Inconsistent subroutine arguments
	Mandatory	IV340	Data handling problem
		IV341	Initialized data incorrectly
		IV342	Accessed or stored data incorrectly
		IV342.1	Flag or index set incorrectly
		IV342.2	Packed/unpacked data incorrectly
		IV342.3	Referenced wrong data variable
		IV342.4	Data referenced out of bounds
		IV343	Scaling or units of data incorrect
		IV344	Dimensioned data incorrectly
		IV344.1	Variable type incorrect

Table 3c-Investigation classification scheme-Type (*Continued*)

Category	Compliance required	Code	Classification
Type IV300		IV344.2	Subscripted variable incorrectly
		IV345	Scope of data incorrect
	Mandatory	IV350	Data problem
		IV351	Sensor data incorrect or missing
		IV352	Operator data incorrect or missing
		IV353	Embedded data in tables incorrect or missing
		IV354	External data incorrect or missing
		IV355	Output data incorrect or missing
		IV356	Input data incorrect or missing
	Mandatory	IV360	Documentation problem
		IV361	Ambiguous statement
		IV362	Incomplete item
		IV363	Incorrect item
		IV364	Missing item
		IV365	Conflicting items
		IV366	Redundant items
		IV367	Confusing item
		IV368	Illogical item
		IV369	Nonverifiable item
	Mandatory	IV370	Unachievable item
	Mandatory	IV380	Document quality problem
		IV381	Application standards not met
		IV382	Not traceable
		IV383	Not current
		IV384	Inconsistencies
		IV385	Incomplete
	Mandatory	IV390	Enhancement
		IV391	Change in program requirements
		IV391.1	Add new capability
		IV391.2	Remove unnecessary capability
		IV391.3	Update current capability
		IV392	Improve comments
		IV393	Improve code efficiency
		IV394	Implement editorial changes
		IV395	Improve usability
		IV396	Software fix of a hardware problem
		IV397	Other enhancement

Table 3c-Investigation classification scheme-Type (*Concluded*)

Category	Compliance required	Code	Classification
IV390/400		IV398	Failure caused by a previous fix
		IV399	Performance problem
		IV400	Interoperability problem
		IV401	Standards conformance problem
		IV402	Other problem

4.2.4.1 Investigation supporting data items

The investigation process, in this context, includes the verification that the anomaly can be reproduced or that it exists, the possible options for dealing with the anomaly, and an analysis of what it would take to implement the possible changes. Table 4 lists some supporting data items that may need documenting during this process. Again, no attempt has been made to make this list exhaustive.

Table 4-Investigation supporting data items

Acknowledgment	Verification
Date received	Source of anomaly
Report number assigned	Data from recognition process
Investigator	
Name	
Code or functional area	
E-mail address	
Phone number	
Estimated start date of investigation	
Estimated complete date of investigation	
Actual start date of investigation	
Actual complete date of investigation	
Person hours	
Date receipt acknowledgment	
Documents used in investigation	
Name	
ID number	
Revision	

4.2.5 Action

Taking action is the next step in the classification process described in figure 1. The minimum classification action required shall be identifying the *Resolution* of the anomaly. The *Resolution* defines how the analyst should choose to deal with the anomaly. An immediate resolution means that the change is implemented immediately, outside of the regular release cycle. The eventual resolution means that the anomaly will be addressed within the regular release cycle, implemented in the release currently in development. The deferred resolution means that the anomaly may be addressed in a future release, preferably one that is in the planning stages.

The *Corrective action* category is optional because not all anomalies considered will require individual corrective actions. When appropriate, such as for an anomaly that can be traced to some procedure and that could be avoided with different procedures, the recommended corrective action should be noted.

Action categories and classification codes all have an AC prefix, and are listed in tables 5a and 5b.

Table 5a-Action classification scheme-Resolution

Category	Compliance required	Code	Classification
Resolution AC100	*Mandatory*	AC110	Immediate
		AC111	Software fix
		AC112	Update project documentation
		AC113	Operator training
		AC114	Testware fix
		AC115	Outside vendor/Third party
	Mandatory	AC120	Eventual
		AC121	Software fix
		AC122	Update project documentation
		AC123	Operator training
		AC124	Testware fix
		AC125	Outside vendor/Third party
	Mandatory	AC130	Deferred
		AC131	Fix in later release
		AC132	Waiver requested (reference)
	Mandatory	AC140	No fix
		AC141	No problem found
		AC142	Waiver requested (reference)
		AC143	Fix not justifiable
		AC144	Fix not identifiable
		AC145	Obsolete

Table 5b-Action classification scheme-Corrective action

Category	Compliance required	Code	Classification
Corrective action AC200	*Optional*	AC210	Department action
		AC211	Revise process (policies/procedures)
		AC212	Implement training
		AC213	Create/Revise/Reinforce standards/ specifications
		AC214	Reallocate people/resources
		AC215	Improve/Enforce audit activities
		AC220	Corporate action
		AC221	Revise process (policies/procedures)
		AC222	Implement training
		AC223	Create/Revise/Reinforce standards/ specifications
		AC224	Reallocate people/resources
		AC225	Improve/Enforce audit activities
		AC230	Industry/Government
		AC231	Sponsor research/education programs
		AC232	Compile/Publish data
		AC233	Create/Revise/Reinforce standards/ specifications
		AC234	Improve/Enforce audit activities
		AC240	Institutions for research/education
		AC241	Research problem
		AC242	Develop new technologies
		AC243	Test alternate approaches
		AC244	Create/Revise tests
		AC245	Enforce educational standards

4.2.5.1 Action supporting data items

In addition to classifying the *Resolution* and *Corrective action*, there are additional details that may be recorded. Potential supporting data items are listed in table 6. These supporting data items are optional.

Table 6-Action supporting data items

Resolution identification	Resolution action	Corrective action
Item to be fixed	Date resolution complete	Standards/Policies/Procedures to be revised
Name	Software	Organization assigned corrective action follow-up
ID number	Documentation	Person assigned corrective action follow-up
Revision	Version/Revision level	Correction action report number
Component(s) within item	Organization assigned to verify resolution	
Name	Person assigned resolution	
ID number		
Revision		
Text describing fix		
Planned date for action items' completion		
Person assigned action items		
Name		
Code or functional area		
E-mail address		
Address		
Phone number		
Planned date of fix completion		
Deferred fix of reference document		
Name		
ID number		
Revision		

4.2.6 Impact

The impact of an anomaly shall be considered at each step of the anomaly process. Impact categories classify the effect upon the product by implementing the modification requested. The *Severity* of an anomaly may be re-evaluated throughout the recognition, investigation, action, and disposition steps of the anomaly process. Identifying the *Severity* of an anomaly is a mandatory category as is identifying the *Project schedule* and *Project cost* impacts of any possible solutions for the anomaly.

Additional categories that may be useful are *Customer value* and *Priority*. The *Customer value* describes the importance to the customer(s) or potential market value of a solution to an anomaly. The *Priority* is a subjective ranking of the anomaly. It takes into account all the other impact categories.

Other impacts that may be applicable to critical or safety system software are the *Mission/Safety*, *Project risk*, *Project quality/Reliability*, and *Societal* impact categories.

Impact Categories and Classification Codes all have an IM prefix, and are listed in tables 7a–7i.

Table 7a-Impact classification scheme-Severity

Category	Compliance required	Code	Classification
Severity IM100	*Mandatory*	IM110	Urgent
		IM120	High
		IM130	Medium
		IM140	Low
		IM150	None

Table 7b-Impact classification scheme-Priority

Category	Compliance required	Code	Classification
Priority IM200	*Optional*	IM210	Urgent
		IM220	High
		IM230	Medium
		IM240	Low
		IM250	None

Table 7c-Impact classification scheme-Customer value

Category	Compliance required	Code	Classification
Customer value IM300	*Optional*	IM310	Priceless
		IM320	High
		IM330	Medium
		IM340	Low
		IM350	None
		IM360	Detrimental

Table 7d-Impact classification scheme-Mission safety

Category	Compliance required	Code	Classification
Mission safety IM400	*Optional*	IM410	Urgent (Prevent completion of mission or jeopardizes personnel safety)
		IM420	High (Adversely affects completion of mission, no workaround solution exists)
		IM430	Medium (Adversely affects completion of mission, workaround solution exists)
		IM440	Low (Inconvenience or annoyance)
		IM450	None of the above

Table 7e-Impact classification scheme-Project schedule

Category	Compliance required	Code	Classification
Project schedule IM500	*Mandatory*	IM510	High
		IM520	Medium
		IM530	Low
		IM540	None

Table 7f-Impact classification scheme-Project cost

Category	Compliance required	Code	Classification
Project cost IM600	*Mandatory*	IM610	High
		IM620	Medium
		IM630	Low
		IM640	None

Table 7g-Impact classification scheme-Project risk

Category	Compliance required	Code	Classification
Project risk IM700	*Optional*	IM710	High
		IM720	Medium
		IM730	Low
		IM740	None

Table 7h-Impact classification scheme-Project quality/reliability

Category	Compliance required	Code	Classification
Project quality/reliability IM800	*Optional*	IM810	High
		IM820	Medium
		IM830	Low
		IM840	None

Table 7i-Impact classification scheme-Societal

Category	Compliance required	Code	Classification
Societal *IM900*	*Optional*	IM910	High
		IM920	Medium
		IM930	Low
		IM940	None

4.2.6.1 Impact supporting data items

Potential supporting data items for recording the impact of an anomaly are listed in table 8.

Table 8-Impact supporting data items

Project impact	
Cost	Analysis
	Fix implementation (Estimated)
	Fix not done (Estimated)
	Resolution (Actual final cost)
Time	Required for fix (Estimated)
	Required for verification (Estimated)
	Impact if fix not done (Estimated)
	Required to implement (Actual final time)
Risk	Text description of risk
Schedule	If resolved
	If not resolved
	Resolution (Actual final schedule)
Contract change	Class I or Class II

4.2.7 Disposition

The final process shown in figure 1 is the disposition step. Throughout the life of an anomaly the *Disposition* category is left blank, which implies that the anomaly is open or active. When an anomaly is disposed of, it shall either be closed or designated as one of three other dispositions-deferred to a later date or release, merged with another anomaly, and referred to another project. Disposition categories and classification codes all have a DP prefix, and are listed in table 9.

Table 9-Disposition classification scheme

Category	Compliance required	Code	Classification
Disposition DP100	*Mandatory*	DP110	Closed
		DP111	Resolution implemented
		DP112	Not a problem
		DP113	Not in scope of project (unresolvable)
		DP114	Outside vendor's problem (reference)
		DP115	Duplicate problem (reference)
		DP120	Deferred (reference)
		DP130	Merged with another problem (reference)
		DP140	Referred to another project (reference)

4.2.7.1 Disposition supporting data items

Supporting data items that may be useful for recording the disposition of an anomaly are listed in table 10.

Table 10-Disposition supporting data items

Anomaly disposition	Verification
Action implemented	Name
Date report closed	Date
Date document update complete	Version/Revision level
Customer notified	Method
Person sending notice	Test case
Date	
Type of notice	
Reference document number (vendor, duplicate, deferred, merged, or waiver anomalies)	

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 ~~anomaly report~~

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.

This annex describes an example for implementing this standard within an anomaly reporting database system. The ability to rapidly retrieve data and construct reports using some or all aspects of the classification scheme is a powerful quality analysis tool. The example presented assumes that the anomaly reporting sys- tem is built upon a relational database management system (RDBMS) that provides the basic functions of querying, paging through each hit from a query (backward and forward, previous and next), adding new records, updating records, deleting records, moving to specified database tables, printing information, and developing reports. Keeping anomaly report data in a relational database facilitates the generation of reports, statistics, and information for trend analysis. Note that there are several commercial products available for anomaly tracking that can be customized to conform to this standard, in addition to the possibility of modifying an existing anomaly tracking database to conform to this standard.

The main RDBMS table contains classifications and supporting data items that generally will not be modified during the anomaly process. Figure A.1 shows the first of two screens for the anomaly input form. The square brackets mark each field in the main table. The *AR Number* (Anomaly Report Number) through *Actual Time* test fields are single entry items. The fields can be updated, but a history of changes is not maintained.

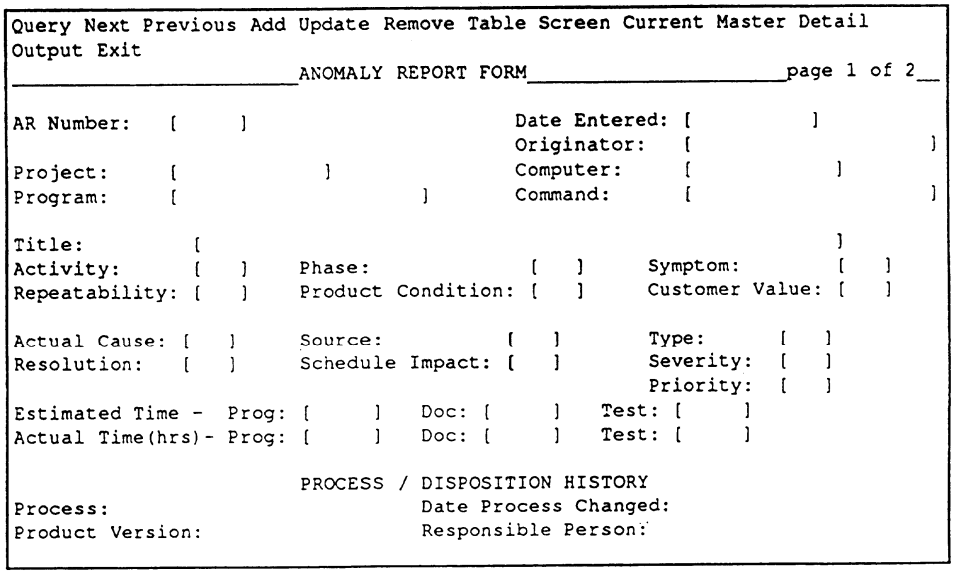
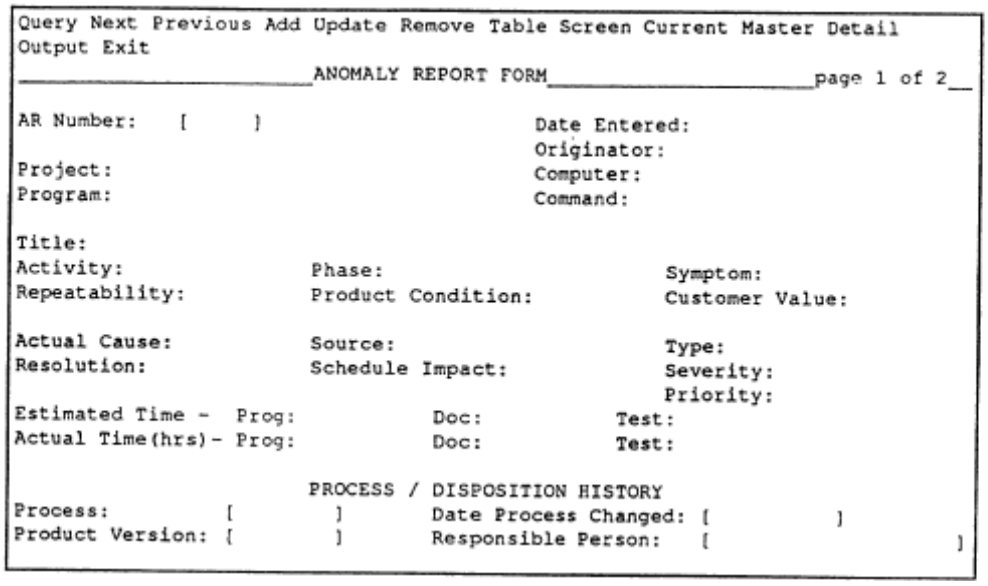


Figure A.1-Example anomaly report-Screen 1, main table

The classification data entered during the recognition process step are *Activity, Phase, Symptom, Repeatability,* and *Customer Value* (*Customer Value* is an impact category). In the form there is room to enter the three- digit code from the standard (minus the two-letter process step identifier). Ideally, the anomaly report system provides the user with the list of the possible classifications and codes for each category.

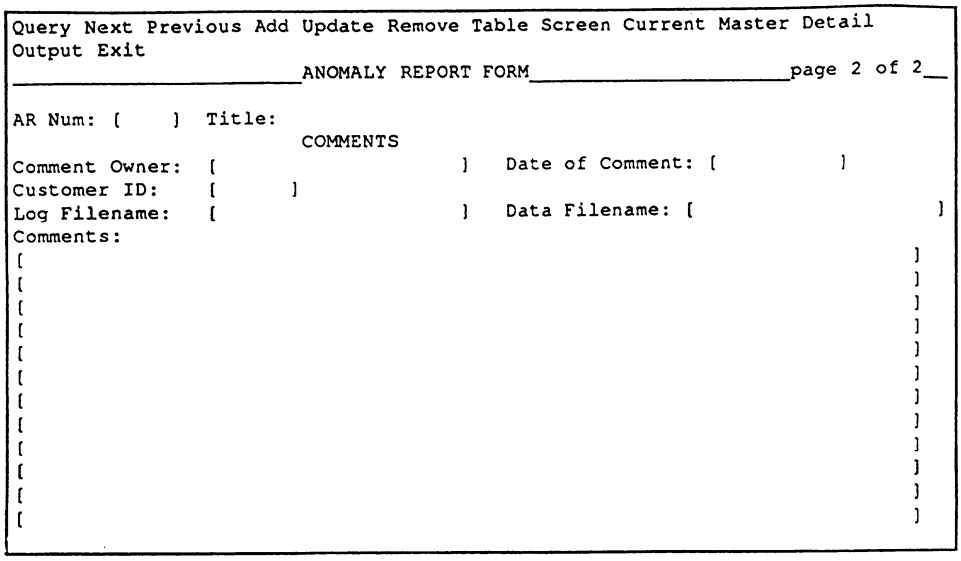
The *Actual Cause, Source, Type, Resolution, Schedule Impact, Severity,* and *Priority* are all categories classified during the investigation through resolution process steps. The estimated time for programming, documenting and testing supporting data items is entered during the investigation process. The actual time for programming, documenting, and testing supporting data items is entered and possibly updated during the action and disposition processes.

The process/disposition history section contains data for maintaining a change history. The purpose of the history RDBMS table is to track the anomalies through the anomaly process steps, and then to identify the disposition chosen for the anomaly. Each time the process step changes, the date of the change, the version number of the affected product, and the person or group responsible for the next action are also recorded. These steps should reflect the anomaly processing mechanism in use for the project. At any time the anomaly's history can be reviewed.



~~Figure A.2-Example anomaly report-Screen 1, history table~~

~~The second screen of the anomaly reporting system (see figure A.3) may also have multiple records relating to the main RDBMS table entry. This screen is used to enter and display comments and other supporting data items for the anomaly. There will probably be comments associated with each step of the anomaly process to provide details on what the anomaly is, how it happened, in what environment it occurred, what should be done about it, what was done about it, suggested corrective actions, and why the particular change involved was implemented. Multiple records in this table associated with an anomaly ensure that all information about an anomaly is maintained.~~



~~Figure A.3-Example anomaly report-Screen 2, comment table~~

~~The *AR Number* and *Title* are displayed from the first screen to provide continuity while entering data. The *Customer ID* field provides a mechanism for tracking anomalies reported by specific customers.~~

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	Failur ID	F080001	F0800	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	V6.4	N/A	V3.4	Initial	V1.0
Defect	Version		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	Functional	N/A	Usability	Functionality	Functionality
Defect	Type	Interface	N/A	Data	Data	Other
Defect	Mode	Missing	N/A	Wrong	Wrong	Missing
Defect	Insertion	Coding	N/A	Coding	Requirement	Design
Defect	Detection	Productio	N/A	Customer	Peer review	Production
Defect	Failure	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.

To obtain previous editions or additional copies of this document, please visit www.techstreet.com or contact Techstreet by phone, email or fax.

Thomson Reuters
Techstreet

3916 Ranchero Drive
Ann Arbor, MI 48108 USA

Phone: +1 734 780 8000
Toll Free: +1 800 699 9277
Fax: +1 734 780 2046

techstreet.service@thomsonreuters.com
techstreet.com

THOMSON REUTERS