IEEE Standard for Information Technology—System and Software Life Cycle Processes—Reuse Processes

IEEE Computer Society

Sponsored by the
Software & Systems Engineering Standards Committee

1517™

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

25 August 2010

IEEE Std 1517™-2010
(Revision of
IEEE Std 1517-1999)

IEEE Std 1517™-2010
(Revision of
IEEE Std 1517-1999)

IEEE Standard for Information Technology—System and Software Life Cycle Processes—Reuse Processes

Sponsor

Software & Systems Engineering Standards Committee
of the
IEEE Computer Society

Approved 17 June 2010

IEEE-SA Standards Board

Approved 25 April 2011

American National Standards Institute

Abstract: A common framework for extending the system and software life cycle processes of IEEE Std 12207™-2008 to include the systematic practice of reuse is provided. The processes, activities, and tasks to be applied during each life cycle process to enable a system and/or product to be constructed from reusable assets are specified. The processes, activities, and tasks to enable the identification, construction, maintenance, and management of assets supplied are also specified.
Keywords: asset, domain engineering, process, reuse, software life cycle, system life cycle

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

PDF: ISBN 978-0-7381-6357-4 STD96085
Print: ISBN 978-0-7381-6358-1 STDPD96085

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 1517-2010, IEEE Standard for Information Technology—System and Software Life Cycle Processes—Reuse Processes.

This standard replaces IEEE Std 1517-1999 [B3].[a] The revision of this standard is a step in the strategy of harmonizing IEEE software and systems engineering standards with those of ISO/IEC JTC 1/SC 7 in order to achieve a fully integrated suite of system and software life cycle processes and guidance for their application.

This revision is closely integrated with IEEE Std 12207-2008.[b] To that end, this revision does not define processes but instead adds tasks and outcomes to processes and activities defined in IEEE Std 12207-2008.

Clause 3 has been modified in this revision to more closely hew to the definitions in Clause 4 of IEEE Std 12207-2008.

Subclause 7.3 of IEEE Std 12207-2008 encompasses many processes specified in IEEE Std 1517-1999, and so those processes have been removed from this revision.

In order to couple more tightly with IEEE Std 122207-2008, the structure of this revision differs from that of IEEE Std 1517-1999. The clauses correspond with clauses of IEEE Std 12207-2008. Clause 5 corresponds to Clause 6 of IEEE Std 12207-2008. Clause 6 corresponds to Clause 7 of IEEE Std 12207-2008.

The objectives defined in Annex C of IEEE Std 1517-1999 have been integrated into the processes of Clause 5 and Clause 6 of this revision. They comprise the reuse-related outcomes specified for the processes.

Software reuse entails capitalizing on existing software and systems to create new products. An organization cannot benefit from reuse by simply creating and employing libraries of assets. Rather, successful reuse requires the integration of reuse-related activities into the life cycle processes used to create the reuse assets associated with software and system development. Unless reuse is explicitly defined in software and system life cycle processes, an organization will not be able to repeatedly exploit reuse opportunities in multiple software projects or products.

Systematic reuse is the practice of reuse according to a consistent, repeatable process. Practicing systematic reuse requires a focus on the use of engineering principles for all reuse assets involved in development. The major benefits that systematic reuse can deliver are as follows:

- Increase software productivity
- Shorten software development and maintenance time
- Reduce duplication of effort
- Move personnel, tools, and methods more easily among projects
- Reduce software development and maintenance costs
- Produce higher quality software products
- Increase software and system dependability

[a] The numbers in brackets correspond to those of the bibliography in Annex A.
[b] Information on references can be found in Clause 2.

— Improve software interoperability and reliability

— Provide a competitive advantage to an organization that practices reuse

There are a variety of approaches to implement the concept of reuse, including systematic and ad hoc reuse. What distinguishes systematic reuse from other methods is the avoidance of multiple versions of otherwise common elements. For example, suppose a reuse approach results in multiple instantiations of a common element. If the instantiations are modified by software developers, then the element is no longer common and can no longer be maintained as a single element. In the context of this standard, systematic reuse excludes such approaches.

The majority of software products can be built with reuse assets—items, such as designs or test plans, that have been designed to be used in multiple contexts. Because reuse assets can apply to software products, implementations of software products, or systems, reuse assets present tremendous opportunity for software reuse.

One major problem encountered by organizations attempting to practice reuse is that reuse is simply missing from their life cycle processes. To harness the benefits of reuse, an organization must incorporate reuse throughout its system and software processes. An organization that creates systems and software first and considers reuse second may not fully benefit from reuse practices. This standard endeavors to address this problem by defining a common framework for reuse activities and by defining how to integrate the practice of reuse into traditional system and software life cycle processes.

Reuse activities describe how software products are built with assets and how to build and manage these assets. The reuse framework presented in this standard covers both the life cycle of a system and the life cycle of a software product.

IEEE Std 12207-2008 establishes a common framework for software and system life cycle processes. This standard provides additional life cycle activities and tasks that augment the practice of systematic reuse. In addition to supplementing the activities defined in IEEE Std 12207-2008, this standard defines outcomes and tasks applicable throughout the life cycle process. Thus, the use of this standard requires access to and understanding of IEEE Std 12207-2008.

For organizations that already employ systematic reuse activities, this standard may be used to determine the conformity of those activities to this standard and as a basis for improvement of those activities where warranted. When establishing or improving systematic reuse activities, organizations are encouraged to assess the business case. Although systematic reuse fosters significant benefits such as those already described, certain costs and risks may prevent benefits of reuse from being fully realized. These factors include the following:

— *The degree to which reuse benefits are relevant to the organization.* For example, a small organization, or an organization that has few resources allocated to developing and maintain software products, may not be in a position to benefit sufficiently from systematic reuse to justify the required commitments and investments.

— *The availability of suitable tools and assets that are designed for reuse.* The capital costs entailed by new software tools can be significant. The costs of acquiring and/or developing assets may not be justified in relation to the expected benefits.

— *The software maturity of the organization.* Although the organization may wish to undertake systematic reuse, its capability maturity may be insufficient to implement the processes in this standard. Moreover, the organization may lack the means to change its infrastructure to support the processes of systematic reuse while continuing to operate its business as usual. Capability maturity should be objectively assessed in relation to this standard, and missing prerequisites, both as to capability maturity and as to infrastructure, need to be put in place before attempting to undertake systematic reuse.

— *The willingness of the people within the organization to make the necessary changes to the way in which they work.* Many software organizations have cultures that are not conducive to systematic reuse. Producing original software is sometimes more well-regarded than reusing existing software. Changing attitudes and associated non-reuse behaviors can be difficult. Policy changes and capital in vestments, which require senior management to be firmly committed to the achievement of systematic reuse, may be necessary.

Organizations interested in undertaking systemic reuse are advised to analyze their abilities to adopt this standard. A business case that clearly describes the goals, investments, costs, risks, and benefits, along with a time line for achieving the transition to systematic reuse, is an excellent way to ensure success.

This standard provides the basis for practices that enable the incorporation of reuse into the system and software life cycle processes.

IEEE Std 1517-2009 may be used to

— Acquire, supply, develop, manage, and maintain reuse assets;

— Acquire, supply, develop, operate, and maintain software products that are built in whole or in part with reuse assets;

— Manage and improve the organization's life cycle processes with respect to the practice of reuse; Establish software management and engineering environments based on reuse activities;

— Foster improved understanding among customers and vendors and parties involved in the reuse-based life cycle of a software product or system and assets;

— Facilitate the use of reuse assets to develop software products and systems;

— Facilitate the development of reuse assets.

IEEE Std 1517-2009 has been written to work with and integrate into IEEE Std 12207-2008.

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 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 Reuse Processes Working Group had the following membership:

Elena Strange, *Chair*

Joshua Brody
Teresa Doran
Lisa Lippincott
James Moore

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

Johann Amsenga
Bakul Banerjee
Juris Borzovs
Pieter Botman
Lyle Bullock
Juan Carreon
Lawrence Catchpole
Michael Chonoles
Keith Chow
Paul Croll
Geoffrey Darnton
David Deighton
Thomas Dineen
Teresa Doran
Scott Duncan
Sourav Dutta
Carla Ewart
Harriet Feldman
Andrew Fieldsend
David Friscia
David Fuschi
Gregg Giesler
Allan Gillard
Lewis Gray
Randall Groves
Louis Gullo
John Harauz
Mark Henley
Frank Hill
Werner Hoelzl
Robert Holibaugh
Bernard Homes
Atsushi Ito
Mark Jaeger
Piotr Karocki
Rameshchandra Ketharaju
Dwayne Knirk
Ronald Kohl
Thomas Kurihara
George Kyle
Marc Lacroix
Susan Land
David J. Leciston
Daniel Lindberg
William Lumpkins
G. Luri
Faramarz Maghsoodlou
Edward McCall
James Moore
Finnbarr Murphy
Michael S. Newman
William Petit
Ulrich Pohl
Gerald Radack
Annette Reilly
Robert Robinson
Terence Rout
Randall Safier
Bartien Sayogo
Robert Schaaf
David J. Schultz
Stephen Schwarm
Gil Shultz
Carl Singer
Luca Spotorno
Friedrich Stallinger
Thomas Starai
Walter Struppler
Marcy Stutzman
Thomas Tullia
David Walden
Paul Work
Oren Yuen
Janusz Zalewski

When the IEEE-SA Standards Board approved this standard on 17 June 2010, it had the following membership:

Robert M. Grow, *Chair*
Richard H. Hulett, *Vice Chair*
Steve M. Mills, *Past Chair*
Judith Gorman, *Secretary*

Karen Bartleson	Young Kyun Kim	Ronald C. Petersen
Victor Berman	Joseph L. Koepfinger*	Thomas Prevost
Ted Burse	John Kulick	Jon Walter Rosdahl
Clint Chaplin	David J. Law	Sam Sciacca
Andy Drozd	Hung Ling	Mike Seavey
Alexander Gelman	Oleg Logvinov	Curtis Siller
Jim Hughes	Ted Olsen	Don Wright

*Member Emeritus

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

Satish K. Aggarwal, *NRC Representative*
Richard DeBlasio, *DOE Representative*
Michael Janezic, *NIST Representative*

Catherine Berger
IEEE Standards Project Editor

Malia Zaman
IEEE Standards Program Manager, Technical Program Development

Contents

IEEE Standard for Information Technology—System and Software Life Cycle Processes—Reuse Processes

1. Overview

1.1 Scope

This standard draws on IEEE Std 12207™-2008 to describe system and software reuse processes.[1] It describes the relationship of reuse processes to system life cycle processes described in Clause 6 (System Life Cycle Processes) of IEEE Std 12207-2008 and software life cycle processes described in Clause 7 (Software Specific Processes) of IEEE Std 12207-2008. The standard defines processes and specifies requirements for the processes but does not specify particular techniques. The scope of the specified processes is broader than that of IEEE Std 12207-2008 because reuse activities transcend the life cycle of any particular system. The specified processes are suitable on an organization-wide basis.

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

1.2 Purpose

Most reuse processes are not distinct from the normal life cycle but instead must be integrated into other life cycle processes. Some acquirers require reuse because it has the potential to achieve faster time-to-market, improved quality, and cost avoidance. These considerations compel a process standard that explains how reuse processes may be incorporated into the life cycle.

1.3 Field of application

This standard applies to not only reuse assets, but also to the acquisition of all software products. It also applies to the acquisition of software services; and to the supply, development, operation, and maintenance of software products, including reuse assets. Off-the-shelf software that is designed for reuse is a special case of assets that is covered by this standard.

The reuse activities specified by this standard are suitable for use on an organization-wide basis.

This standard is written for acquirers of software products—including assets—and services; and for suppliers, developers, operators, maintainers, managers, quality assurance managers, reuse program administrators, asset managers, domain engineers, and users of software products and assets. Acquirers and suppliers may be internal or external to the organization.

1.3.1 Limitations

The limitations of this standard are in

- Describing a high-level framework for reuse activities, but not the details of how to perform the activities;
- Describing the responsibilities inherent in the various activities, but not the detailed data or control connections among the activities;
- Specifying system and software life cycle reuse activities at the reference level, but not prescribing a specific life cycle process model or methodology;
- Viewing reuse activities as an assignment of continuing responsibilities to agents, but not as a series of steps (procedures) to be performed;
- Specifying provisions for acquiring reuse assets, but not provisions for integration of commercial off- the-shelf (COTS) assets that are not supplied in source code form.

In this standard, as in IEEE Std 12207-2008, there are a number of lists for tasks. None of these is presumed to be exhaustive; they are intended as examples.

1.4 Conformance

This standard adds outcomes and tasks to the existing processes of IEEE Std 12207-2008. Moreover, this standard specifies activities in addition to those of IEEE Std 12207-2008. Conformance to this standard requires conformance to IEEE Std 12207-2008.

An organization may claim “selective conformance” to a designated process by implementing, via both plans and performance, all of the requirements specified as mandatory (by the word *shall*) in the activities and tasks of that process. In the case where this standard references a process specified in IEEE Std 12207-

2008, then conformance requires implementation of the corresponding requirements of IEEE Std 12207-2008 as well as the requirements of this standard.

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

IEEE Std 12207-2008, Systems and software engineering—Software life cycle processes. [2, 3, 4]

3. Definitions

For the purposes of this document, the following terms and definitions apply. *The IEEE Standards Dictionary: Glossary of Terms & Definitions* should be referenced for terms not defined in this clause.[5] ISO/IEC 24765:2009 [B8] may be referenced for additional terms.[6, 7]

acquirer: Stakeholder that acquires or procures a product or service from a supplier. The acquirer could be one of the following: buyer, customer, owner, purchaser.

acquisition: Process of obtaining a system, software product, or software service.

activity: Set of cohesive tasks of a process. (IEEE Std 12207-2008)

application engineering: The processes of constructing or refining application systems by reusing assets.

assemble: The process of constructing from parts one of more identified pieces of software.

classification: The manner in which the assets are organized for ease of search and extraction within a reuse library.

construction: The process of writing, assembling, or generating assets.

developer: Organization that performs development tasks (including requirements analysis, design, testing through acceptance) during a life cycle process.

domain: A problem space.

[2] IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org).
[3] The IEEE standards or products referred to in Clause 2 are trademarks owned by the Institute of Electrical and Electronics Engineers, Incorporated.
[4] ISO/IEC 12207:2008(E) is identical to IEEE Std 12207-2008. Therefore, all references to IEEE Std 12207-2008 apply equally well to their ISO/IEC counterpart.
[5] *The IEEE Standards Dictionary: Glossary of Terms & Definitions* is available at http://shop.ieee.org/.
[6] An additional source is available at http://pascal.computer.org/sev_display/index.action.
[7] The numbers in brackets correspond to those of the bibliography in Annex A.

domain architecture: A generic, organizational structure or design for software systems in a domain. The domain architecture contains the designs that are intended to satisfy requirements specified in the domain model. The domain architecture documents design, whereas the domain model documents requirements. A domain architecture: 1) can be adapted to create designs for software systems within a domain, and 2) provides a framework for configuring assets within individual software systems.

NOTE—The term *architecture* has been deliberately refined from its current IEEE standard definition to more properly convey its meaning in the software reuse context.[8]

domain engineer: A party that performs domain engineering activities (including domain analysis, domain design, asset construction, and asset maintenance).

domain engineering: A reuse-based approach to defining the scope (i.e., domain definition), specifying the structure (i.e., domain architecture), and building the assets (e.g., requirements, designs, software code, documentation) for a class of systems, subsystems, or applications. Domain engineering may include the following activities: domain definition, domain analysis, developing the domain architecture, and domain implementation.

domain expert: An individual who is intimately familiar with the domain and can provide detailed information to the domain engineers.

domain model: A product of domain analysis that provides a representation of the requirements of the domain. The domain model identifies and describes the structure of data, flow of information, functions, constraints, and controls within the domain that are included in software systems in the domain. The domain model describes the commonalities and variabilities among requirements for software systems in the domain.

life cycle: Evolution of a system, product, service, project, or other human-made entity from conception through retirement.

maintainer: Organization that performs maintenance activities.

reusability: (A) The degree to which an asset can be used in more than one software system or in building other assets. **(B)** In a reuse library, the characteristics of an asset that make it easy to use in different contexts, software systems, or in building different assets.

reuse: The use of an asset in the solution of different problems.

[reuse] asset: An item, such as design, specification, source code, documentation, test suites, manual procedures, etc., that has been designed for use in multiple contexts.

NOTE—This term has been deliberately redefined from its current IEEE standard definition to more properly convey its meaning in the software reuse context.

reuse sponsor: A member of the organization's management who authorized, approves, promotes, and obtains the funding and other resources for the Reuse Program.

software item: Source code, object code, control data, or a collection of these items.

software unit: Separately compileable piece of code.

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

system: A combination of interacting elements organized to achieve one or more stated purposes. A system may be considered as a product or as the services it provides. In practice, the interpretation of its meaning is frequently clarified by the use of an associative noun, e.g., aircraft system. Alternatively, the word *system* may be substituted simply by a context-dependent synonym, e.g., aircraft, though this may then obscure a system principles perspective. (adapted from IEEE Std 12207-2008)

systematic reuse: The practice of reuse according to a well-defined, repeatable process.

template: An asset with parameters or slots that can be used to construct an instantiated asset. *See also:* **construction.**

usability: A measure of an executable software unit's or system's functionality, ease of use, and efficiency. *See also:* **reusability.**

NOTE—This term has been deliberately redefined from its current IEEE standard definition to more properly convey its meaning in the software reuse context.

user: An individual or group that benefits from a system during its utilization.

4. Application of this standard

This clause presents the framework of the reuse life cycle activities that can be employed to

- Develop, operate, and maintain systems and software products with reuse assets;
- Develop and maintain reuse assets.

4.1 Organization of this standard

4.1.1 Reuse activities and tasks

This standard specifies the system life cycle and software life cycle activities and tasks that are required to practice reuse. They fall into the following four reuse categories:

- Development, operation, and maintenance of products with reuse assets
- Development and maintenance of reuse assets
- Management of the practice of reuse
- Management of reuse assets

This standard uses the software and system life cycle frameworks described in IEEE Std 12207-2008 as the foundation for describing a life cycle that includes the practice of reuse. Reuse activities and tasks are integrated into the IEEE 12207™ framework.

Each life cycle process discussed in IEEE Std 12207-2008 is divided into a set of activities, and the requirements of each activity are specified in a set of tasks. This standard adds tasks to the activities defined in IEEE Std 12207-2008, where appropriate for reuse. Subclause number x.y.z denotes a process, v.w.x.y.z denotes an activity, and u.v.w.x.y.z denotes a task. Correlation of the processes, activities, and tasks in this standard with those in IEEE Std 12207-2008 is based on the name, not the number, associated with a process or activity. Numbers of activities and tasks specified in this standard are not intended to directly map to activities and tasks specified in IEEE Std 12207-2008. When a task defined in this standard

adds requirements to an existing task in IEEE Std 12207-2008, the task number of the affected IEEE 12207 tasks is indicated in the text for the corresponding reuse task specified in this standard.

4.1.2 Development of system products and services

Reuse activities that are related to the acquisition, infrastructure, implementation, maintenance, and disposal of systems are integrated into Clause 6 of IEEE Std 12207-2008. With the addition of these reuse activities and tasks, the system life cycle processes described in this standard employ domain models, domain architectures, and reuse assets to develop and maintain products and services.

The acquirer, supplier, developer, and maintainer are the parties or agents responsible for performing the system life cycle process.

4.1.3 Development of software products

Reuse activities and tasks that are related to the development, operation, and maintenance of software products with reuse assets are integrated into Clause 7 of IEEE Std 12207-2008. With the addition of these reuse activities and tasks, the software life cycle processes described in this standard employ domain models, domain architectures, and reuse assets to develop and maintain software products.

The acquirer, supplier, developer, operator, and maintainer are the parties or agents responsible for performing the software life cycle process.

4.1.4 Management of the practice of reuse

This standard groups the activities and tasks that fall into the third reuse category (i.e., the management of the practice of reuse) into the Reuse Program Administration Process. This process is specified as an organizational life cycle process: a process employed by an organization to establish and implement an underlying structure. By specifying Reuse Program Administration as an organizational process, reuse can be practiced in a systematic manner within a single project as well as across an organization. Typically, the Reuse Program Administration Process is employed outside the realm of a specific project or contract in order to manage and coordinate the practice of systematic reuse across multiple projects or contracts in an organization.

The reuse program administrator is the party or agent responsible for perform the Reuse Program Administration Process.

5. Integration of reuse into system life cycle processes

This clause defines the reuse activities and tasks to implement systematic reuse in order to productively develop, operate, and maintain standalone software products or services or software systems. These reuse activities and tasks are integrated into the following activities and tasks that have been defined in the IEEE 12207 system life cycle processes:

a) Agreement Processes

b) Organizational Project-Enabling Processes

c) Project Processes

d) Technical Processes

5.1 Agreement Processes

To assure that the Agreement Processes support reuse activities, the acquirer shall perform the following additional reuse-related tasks when performing these IEEE 12207 Agreement Processes:

a) Acquisition Process

b) Supply Process

5.1.1 Acquisition Process

5.1.1.1 Outcomes

The following reuse-related outcomes are added to this process:

a) The acquirer identifies existing and soon-to-be-available assets that may be appropriate for the project

b) Reuse is considered as an option for acquisition as part of the analysis of risk, cost, and benefit criteria

c) An existing acquisitions plan template is reused, if an appropriate one is available, to create the acquisition plan

d) The acquisition document includes any requirements for practicing reuse that the supplier will be required to meet

5.1.1.2 Activities and tasks

5.1.1.2.1 Acquisition Preparation

The following reuse-related tasks are added to this activity:

5.1.1.2.1.1 Before acquiring software, the acquirer shall review current, available, and soon-to-be-available assets to determine if what is needed exists or can be assembled to develop the needed software. The acquirer may task the supplier, domain engineer, or reuse asset manager to identify reuse assets for consideration.

5.1.1.2.1.2 The acquirer shall require reuse assets to

a) Have interfaces compatible with the domain architecture;

b) Be compatible with the domain model.

5.1.1.2.1.3 The acquirer shall determine whether reuse is suitable for the project and the organization. If appropriate, the acquirer shall consider and document reuse as an option for acquisition as part of the acquirer's analysis of risk, cost, and benefit criteria. The acquirer shall compare software requirements with reuse asset capabilities and, if proper, evaluate and document costs and benefits of modifying software requirements to enable or increase the reuse of assets. (This task defines reuse-related requirements in addition to those requirements specified in IEEE Std 12207-2008, task 6.1.1.3.1.6.)

5.1.1.2.1.4 When an off-the-shelf software product or asset is to be acquired, the acquirer should select products or assets that meet the standards and criteria for reusability of the acquirer's organization. (This task defines reuse-related requirements in addition to those requirements specified in IEEE Std 12207-2008, task 6.1.1.3.1.7.)

5.1.1.2.1.5 The acquirer shall create and document an acquisition plan, reusing an applicable acquisition plan template, if any exists, to define the resources and procedures to acquire software. The acquirer may use the acquisition plan template included in Annex B of IEEE Std 1067™-2005 [B2]. (This task defines reuse-related requirements specified in IEEE Std 12207-2008, task 6.1.1.3.1.8.)

5.1.1.2.2 Acquirer acceptance

The following reuse-related task is added to this activity:

5.1.1.2.2.1 If the software product is to be reusable then acceptance test cases should include reusability tests and domain architecture compatibility tests. (This task defines reuse-related requirements in addition to those requirements specified in IEEE Std 12207-2008, task 6.1.1.3.6.2.)

5.1.2 Supply Process

5.1.2.1 Outcomes

The following outcomes are added to this process:

a) The agreement between the acquirer and supplier is sufficiently documented and available to the supplier for future projects

b) The agreement is prepared using a reusable proposal template, if an appropriate one exists

c) A contact is prepared using a reusable contract template, if an appropriate one exists

5.1.2.2 Activities and tasks

5.1.2.2.1 Supplier tendering

The following reuse-related task is added to this activity:

5.1.2.2.1.1 The supplier shall determine whether a proposal exists that can be modified to satisfy the agreed requirements.

5.1.2.2.1.2 The supplier shall modify an existing proposal, if appropriate, to satisfy the agreed requirements.

5.1.2.2.2 Contract agreement

The following reuse-related task is added to this activity:

5.1.2.2.2.1 The supplier shall determine whether a life cycle model appropriate to the scope, magnitude, and complexity of the project already exists (in the context of the acquirer) or can be assembled to develop the contract. The supplier may task the acquirer to identify life cycle models for consideration.

5.1.2.2.3 Contract execution

The following reuse-related task is added to this activity.

5.1.2.2.3.1 In addition to the options enumerated in IEEE 12207 task 6.1.2.3.4.4, the supplier shall consider the following options for developing the software product or providing the software service:

a) Reuse an existing software product from internal resources

b) Reuse an existing software product provided by a subcontractor

5.2 Organizational Project-Enabling Processes

To assure that the system life cycle processes support reuse activities, the organization shall perform the following additional reuse-related tasks when performing these IEEE 12207 Organizations Project-Enabling Processes:

a) Life Cycle Model Management Process

b) Infrastructure Management Process

c) Project Portfolio Management Process

d) Quality Management Process

5.2.1 Life Cycle Model Management Process

5.2.1.1 Outcomes

The following reuse-related outcomes are added to this process:

a) An asset management plan is developed and documented reusing an applicable asset management plan template, if any exists

b) A reuse asset classification scheme is developed, documented, and maintained

c) For each reuse asset submitted, the asset is evaluated according to reuse criteria and procedures

d) For each reuse asset accepted, the asset is classified according to reuse classification schemes and made available through a reuse asset storage and retrieval mechanism

5.2.1.2 Activities and tasks

5.2.1.2.1 Process establishment

The follow reuse-related task is added to this activity:

5.2.1.2.1.1 The organization shall consider the degree to which reuse benefits are relevant to the organization. If few resources are allocated to the development and maintenance of software products, the organization may not be in a position to implement all reuse-related tasks associated with this activity.

5.2.1.2.1.2 The organization shall determine whether a suite of organizational processes for software life cycle and processes and models exists or can be modified for its purposes.

5.2.1.2.2 Process assessment

5.2.1.2.2.1 The organization shall identify existing process assessment records to determine if a processes assessment can be reused.

5.2.1.2.2.2 The process assessment shall include assessment of reusability tasks.

5.2.2 Infrastructure Management Process

5.2.2.1 Outcomes

The following outcome is added to this process:

a) Existing infrastructure documentation is applied or modified for the project.

5.2.2.2 Activities and tasks

5.2.2.2.1 Process implementation

The following reuse-related task is added to this activity.

5.2.2.2.1.1 Existing infrastructure documentation should be reviewed to determine whether it meets the requirements of the process employing it. If possible, infrastructure documentation should be applied or modified, considering the applicable procedures, standards, and techniques of the project.

5.2.2.2.2 Maintenance of the infrastructure

The following reuse-related task is added to this activity:

5.2.2.2.2.1 When modifications to infrastructure are implemented as part of the maintenance process, they shall be recorded so that all other projects that reuse the infrastructure can be examined to determine the impact these modifications may have.

5.2.3 Project Portfolio Management Process

5.2.3.1 Outcomes

5.2.3.1.1 Project initiation

The following reuse-related task is added to this activity.

5.2.3.1.1.1 The organization shall determine whether it can benefit from reuse of existing project reporting requirements, given the initial outlay of resources necessary to review them. If applicable, the organization shall reuse existing project reporting requirements.

5.2.3.1.2 Portfolio evaluation

The following reuse-related task is added to this activity:

5.2.3.1.2.1 In addition to the tasks defined in IEEE Std 12207-2008, the organization shall evaluate ongoing projects to confirm that

a) Projects are complying with reuse directives.

5.2.4 Quality Management Process

5.2.4.1 Outcomes

The following reuse-related outcome is added to this process:

a) Existing organization quality objectives and policies are used or modified for the project, if applicable

5.2.4.2 Activities and tasks

5.2.4.2.1 Quality management

The following reuse-related tasks are added to this activity:

5.2.4.2.1.1 The organization shall determine if existing quality management policies, standards, and procedures can be used or modified for the project.

5.2.4.2.1.2 The organization shall use existing quality management goals and objectives based on business strategy for customer satisfaction.

5.2.4.2.1.3 The organization shall determine whether customer satisfaction criteria are established and applicable to the project. When assessing customer satisfaction, the organization shall document its criteria for future reuse purposes.

5.3 Project Processes

To assure that the Project Processes support the reuse activities, the manager shall perform additional reuse-related tasks when performing the following IEEE 12207 Project Processes:

a) Project Planning Process

b) Project Assessment and Control Process

c) Decision Management Process

5.3.1 Project Planning Process

5.3.1.1 Activities and tasks

5.3.1.1.1 Project planning

The following reuse-related tasks are added to this activity:

5.3.1.1.1.1 When defining or selecting a life cycle model for this supply project, the supplier shall evaluate and document the capabilities of this model to satisfy reuse process requirements. The supplier should explore with the acquirer the possibility of modifying software requirements to enable or increase the reuse of assets and, if proper, evaluate and document the costs and benefits thereof.

5.3.1.1.1.2 Once the planning requirements are established, the supplier shall consider and document the option of developing the system, software product, or asset with reuse assets as part of the supplier's analysis of risks and benefits associated with this option.

5.3.1.1.1.3 Reuse strategies that the supplier shall consider and document in the project plan include the following:

a) Developing the software product with reuse assets

b) Developing the software product, or some parts of it, as a reuse asset

5.3.1.1.1.4 The supplier should identify and document, in the project plans, candidate software products and reuse assets that may be developed within the scope and context of this project. This information should be communicated using a domain engineering feedback mechanism and a reuse asset management communication mechanism.

5.3.2 Project Assessment and Control Process

5.3.2.1 Purpose

The Project Assessment and Control Process shall include reuse-related measures when determining the status of the project. To be satisfactory, the project must perform according to plans and schedules, within projected budgets, and satisfy technical objectives and reuse objectives.

5.3.2.2 Outcomes

The following reuse-related outcome is added to this process:

a) Exploitation of reusable assets is monitored as an essential component of progress of the project

5.3.2.3 Activities and tasks

5.3.2.3.1 Project monitoring

The following reuse-related task is added to this activity:

5.3.2.3.1.1 The manager shall monitor the reuse-related activities of the project.

5.3.2.3.2 Project control

The following reuse-related tasks are added to this activity:

5.3.2.3.2.1 When problems are discovered, the manager shall determine whether similar problems have surfaced in other projects, and whether those other resolutions are applicable to the project.

5.3.2.3.2.2 If the resolution of a problem results in a change to plans, the manager shall determine whether current, available, or soon-to-be-available assets, investigated as part of the Acquisition Preparation Process (5.1.1.2.1) satisfy the new plan.

5.3.2.3.2.3 The manager shall include advancement toward reusability goals in internal progress reports.

5.3.3 Decision Management Process

5.3.3.1 Activities and tasks

5.3.3.1.1 Decision planning

The following reuse-related activities are added to this task:

5.3.3.1.1.1 Where appropriate, the project shall reuse existing decision-making strategies.

5.3.3.1.1.2 The project shall document relevant decision-making strategies for reuse in future projects.

5.3.3.1.1.3 The project shall review records of problems and opportunities and their solutions to determine whether they shall be applicable in future projects.

5.3.4 Risk Management Process

5.3.4.1 Outcomes

The following reuse-related outcome is added to the process:

a) Reuse is considered a component in the analysis of risk management

b) The reuse and risk management policies of other projects are evaluated

5.3.4.2 Activities and tasks

5.3.4.2.1 Risk management planning

The following reuse-related tasks are added to this activity:

5.3.4.2.1.1 Where appropriate, risk management policies from other projects shall be reused or adapted.

5.3.4.2.1.2 The Risk Management Process description shall be documented in a way that it can be reused or adapted for future projects. The stakeholders shall distinguish specific from general risks and document the management of these risks appropriately.

5.3.4.2.2 Risk profile management

The following reuse-related task is added to this activity:

5.3.4.2.2.1 In addition to the risk profile requirements described in IEEE Std 12207-2008, 6.3.4.3.2.3, the risk profile shall include a record of risk's state in previous projects, if applicable.

5.3.4.2.3 Risk analysis

5.3.4.2.3.1 The probability of occurrence and consequences of each risk identified shall be estimated, if applicable, by assessing the outcomes of similar risks in previous projects.

5.3.5 Configuration Management Process

5.3.5.1 Activities and tasks

5.3.5.1.1 Configuration management planning

The following reuse-related task is added to this activity:

5.3.5.1.1.1 Before defining a configuration management strategy, the project shall review and consider configuration management strategies from other projects.

5.3.6 Information Management Process

5.3.6.1 Activities and tasks

5.3.6.1.1 Information management planning

The following reuse-related task is added to this activity:

5.3.6.1.1.1 Where applicable, the project shall document the items of information relevant for reuse in future projects.

5.3.6.1.2 Information management execution

The following reuse-related task is added to this activity:

5.3.6.1.2.1 The project shall make designated information archives available to future projects for reuse considerations.

5.3.7 Measurement Process

5.3.7.1 Activities and tasks

5.3.7.1.1 Measurement planning

The following reuse-related tasks are added to this activity:

5.3.7.1.1.1 When describing characteristics of the organization that are relevant to measurement, the project shall review such characteristics documented from previous projects.

5.3.7.1.1.2 When documenting measures that satisfy information needs, the project shall consider and document such measures that may apply to future projects.

5.3.7.1.1.3 Where applicable, the project shall reuse existing data collection, analysis, and reporting procedures. When a new procedure is used, the project shall document the procedure in such a way that it can be useful for future projects.

5.4 Technical Processes

To assure that the system life cycle processes support reuse activities, the project shall perform the following additional reuse-related tasks when performing these IEEE 12207 Technical Processes activities:

a) Stakeholder Requirements Definition Process

b) System Requirements Analysis Process

c) System Architectural Design Process

d) System Integration Process

e) System Qualification Testing Process

f) Software Installation Process

g) Software Acceptance Support Process

h) Software Operation Process

i) Software Maintenance Process

j) Software Disposal Process

5.4.1 Stakeholder Requirements Definition Process

5.4.1.1 Activities and tasks

5.4.1.1.1 Stakeholder identification

The following reuse-related task is added to this activity:

5.4.1.1.1.1 Where applicable, the project shall identify stakeholders in similar positions from previous projects.

5.4.1.1.2 Requirements identification

The following reuse-related task is added to this activity:

5.4.1.1.2.1 Where applicable, the project shall consider reusing existing stakeholder requirements documentation.

5.4.1.1.3 Requirement recording

The following reuse-related task is added to this activity:

5.4.1.1.3.1 When recording requirements, the project shall make such documentation available for future projects.

5.4.2 System Requirements Analysis Process

5.4.2.1 Outcomes

The following reuse-related outcome is added to this process:

a) Existing reusable system requirements, if applicable, are considered to create the system requirements specifications

5.4.2.2 Activities and tasks

5.4.2.2.1 Requirements specification

The following reuse-related tasks are added to this activity:

5.4.2.2.1.1 The developer shall consider applicable system requirements specifications, if any exist, before writing new requirements specifications. When writing new requirements specifications, the developer shall use the language and concepts of the domain models for the domain to which this software system belongs.

5.4.2.2.1.2 The systems requirements specification shall describe system reusability requirements, including the domain architectures interface requirements.

5.4.2.2.2 Requirements evaluation

5.4.2.2.2.1 The system requirements specification shall be evaluated according to reusability criteria. The results of the evaluation shall be documented.

NOTE—Reusability criteria include, but are not limited to the following:

a) Usability and reusability of the system requirements

b) Identification of customized requirements that can be replaced by reused requirements if the acquisition needs are adjusted

c) Reuse potential of the requirements to be used in multiple contexts

5.4.2.2.2.2 The developer shall use a domain engineering feedback mechanism and a reuse asset management communication mechanism to report reuse information about requirements specifications.

NOTE—Reuse information includes, but is not limited to the following:

a) Missing assets for the domain

b) Nominally reusable assets for the domain rejected for this project and the rejection reasons

c) Suggested modifications to reuse assets

d) List of assets reused by this project

e) List of new candidate reuse assets suggested by this project

5.4.3 System Architectural Design Process

5.4.3.1 Outcomes

The following reuse-related outcome is added to this process:

a) Reusability is considered in the evaluation of the system architecture.

5.4.3.2 Activities and tasks

5.4.3.2.1 Establishing architectures

The following reuse-related task is added to this activity:

5.4.3.2.1.1 The system architecture shall be derived from the existing system architectures, if applicable.

5.4.3.2.2 Architectural evaluation

The following reuse-related task is added to this activity:

5.4.3.2.2.1 The system architecture shall be evaluated according to reusability criteria. The results of the evaluation shall be documented.

NOTE—Reusability criteria include, but are not limited to the following:

a) Usability and reusability of the system architecture and its parts

b) Reuse potential of the system architecture or its parts to be used in multiple contexts

c) Compatibility of the system architecture with the domain models and domain architectures

5.4.3.2.2.2 If appropriate, the developer shall modify the system requirements in response to this evaluation. The developer shall re-analyze the system requirements as described in 5.4.2 of this standard.

5.4.4 System Integration Process

5.4.4.1 Activities and tasks

5.4.4.1.1 Test readiness

The following reuse-related tasks are added to this activity:

5.4.4.1.1.1 For each qualification requirement of the system, a set of tests, test cases, and test procedures shall be developed, reusing applicable test assets, if any exist.

5.4.4.1.1.2 The integrated system shall be evaluated according to reusability criteria. The results of the evaluation shall be documented.

NOTE—Reusability criteria include, but are not limited to the following:

a) Compliance to the organization’s reuse standards

b) Usability and reusability of the system and qualification tests

c) Reuse potential of the system, its parts, and its qualification tests to be used in multiple contexts

5.4.5 System Qualification Testing Process

5.4.5.1 Outcomes

The following reuse-related outcome is added to this process:

a) System qualification testing is performed reusing applicable qualification tests, if any exist

5.4.5.2 Activities and tasks

5.4.5.2.1 Qualification testing

The following reuse-related task is added to this activity:

5.4.5.2.1.1 The system shall be evaluated according to reusability criteria. The results of the evaluation shall be documented.

NOTE—Reusability criteria include, but are not limited to the following:

a) Usability and reusability of the software portion of the system

b) Reuse potential of the software parts for he system to be used in multiple contexts

5.4.6 Software Installation Process

5.4.6.1 Activities and tasks

5.4.6.1.1 Software installation

The following reuse-related tasks are added to this activity:

5.4.6.1.1.1 The developer shall create and document an installation plan, reusing an applicable installation plan template, if any exists, to define the resources and procedures to install the software product on its target environment(s).

5.4.6.1.1.2 The installation plan and installation plan template shall be evaluated according to reusability criteria. The results of the evaluation shall be documented.

NOTE—Reusability criteria include, but are not limited to the following:

a) Usability and reusability of the installation plan and installation plan template

b) Reuse potential of the installation plan to be used in multiple contexts

5.4.7 Software Acceptance Support Process

5.4.7.1 Activities and tasks

5.4.7.1.1 Software acceptance support

The following reuse-related task is added to this activity:

5.4.7.1.1.1 The acquirer shall consider reuse as a condition of software acceptance. Where appropriate, the acquirer shall document this condition as a consideration for future projects.

5.4.8 Software Operation Process

5.4.8.1 Activities and tasks

5.4.8.1.1 Preparation for operation

The following reuse-related tasks are added to this activity:

5.4.8.1.1.1 The operator shall use existing plans and operational standards, if applicable.

5.4.8.1.1.2 When documenting testing procedures, the operator shall consider the reuse applications of such procedures in future projects.

5.4.9 Software Maintenance Process

5.4.9.1 Activities and tasks

5.4.9.1.1 Process implementation

The following reuse-related task is added to this activity:

5.4.9.1.1.1 The maintainer shall develop a maintenance plan reusing an applicable maintenance plan template, if any exists.

5.4.9.1.2 Problem and modification analysis

The following reuse-related tasks are added to this activity:

5.4.9.1.2.1 The maintainer shall establish procedures for receiving, recording, resolving, and tracking problems, and providing feedback to the asset manager whenever problems are encountered in or change requests are made for reuse assets that were reused in the development of the software product.

5.4.9.1.2.2 Whenever problems with or change requests for reuse assets are encountered, they shall be recorded in accordance with Process 7.2.8, Problem Resolution Process, of IEEE Std 12207-2008, so that all other software products that reused these reuse assets can be examined to determine the impact that these problems or change requests may have on these software products.

5.4.9.1.3 Migration

The following reuse-related tasks are added to this activity:

5.4.9.1.3.1 When a system that has been developed from assets is to be migrated to a new environment, migration planning activities shall include notification of the asset manager.

5.4.9.1.3.2 After a system that has been constructed from assets has been migrated to a new environment, notifications shall be sent to the asset manager.

5.4.9.1.3.3 When the migrated system has been constructed from assets, the results of a post-operation review that assesses the impact of changing to the new environment shall be sent to the asset manager and the domain engineer.

5.4.10 Software Disposal Process

5.4.10.1 Activities and tasks

5.4.10.1.1 Software disposal planning

The following reuse-related task is added to this activity:

5.4.10.1.1.1 In addition to the items listed in IEEE Std 12207-2008, 6.3.11.3.1.1, the software disposal plan shall address the following:

a) Notification of the asset manager

5.4.10.1.2 Software disposal execution

The following reuse-related tasks are added to this activity:

5.4.10.1.2.1 When a system that has been constructed from reuse assets is to be retired, retirement planning activities shall include notifying the asset manager.

5.4.10.1.2.2 When a system constructed from reuse assets is retired, notifications shall be sent to the asset manager.

6. Integration of reuse into software-specific life cycle processes

This clause defines the reuse activities and tasks required to develop, operate, and maintain standalone software products or services or elements of software systems. These reuse activities and tasks are integrated into the following IEEE 12207 software life cycle processes as described in this clause:

a) Software Implementation Processes

b) Software Support Processes

c) Software Reuse Processes

6.1 Software Implementation Processes

The Software Implementation Processes contain the activities and tasks of the developer. In the context of this standard, the process contains the activities for implementation, requirements analysis, architectural design, detailed design, construction, integration, and qualification testing related to software products developed with assets. The Development Process based on developing a software product with assets is also referred to as *application engineering.*

When the reuse-based life cycle is deployed to build software products with assets, the assets employed should be designed for use in multiple contexts. Reuse assets should include the domain architecture and assets that have been built for the domain in which the software product is a member. In addition, the assets used to build this software product may be obtained from other resources including other domain engineering processes, the organization's reuse libraries, other ongoing internal projects, and outside suppliers.

The developer must weigh the benefits of systematic reuse against the costs of risks of following reuse processes. A developer might benefit from the reuse of a software component if, for example, it has been sufficiently tested and used in other projects. Moreover, a developer might benefit from reuse of an existing software component if it is modular and easy to modify. The developer must take these factors into consideration when investigating the outcomes and tasks defined for the following processes.

Software Implementation Processes, based on developing a software product with reuse assets, consist of the following processes:

a) Software Implementation Process

b) Software Requirements Analysis Process

c) Software Architectural Design Process

d) Software Detailed Design Process

e) Software Construction Process

f) Software Integration Process

g) Software Qualification Testing Process

6.1.1 Software Implementation Process

6.1.1.1 Outcomes

The following reuse-related outcomes are added to this process:

a) A software item is realized with reusable components isolated and documented

6.1.1.2 Activities and tasks

6.1.1.2.1 Software implementation strategy

The following reuse-related tasks are added to this activity:

6.1.1.2.1.1 The developer shall evaluate and document the capabilities of the model to satisfy reuse requirements when selecting or defining a life cycle process model appropriate to the scope, magnitude, and complexity of this project. The reuse-related activities and tasks shall be selected and mapped onto the life cycle process model chosen.

6.1.1.2.1.2 The developer shall select and use standards, methods, tools, and computer programming languages that enable, support, and enforce the practice of reuse in the project.

6.1.1.2.1.3 The developer shall create and document a project plan, reusing an applicable project plan template, if any exists, to define the resources and procedures to develop the software. The plans should include specific standards, methods, tools, and programming languages that have been selected to enable the practice of reuse in the project.

6.1.1.2.1.4 The developer shall utilize the following project mechanisms:

a) Feedback mechanism to the domain engineer to communicate the use and impact of software products and reuse assets on this project

b) Communication mechanism with the asset manager to resolve problems, answer questions, and make recommendations concerning software products and assets that this project encounters

c) Notification mechanism that makes the developer aware of the prevailing trade laws, the licensing properties of software products and assets, the organization's restrictions that protect its proprietary interests, and the contract that may restrict or exclude the use of specific software products or assets by this project

d) Notification mechanism that informs the developer about any changes or problems or other useful information concerning relevant assets

6.1.2 Software Requirements Analysis Process

6.1.2.1 Outcomes

The following reuse-related outcome is added to this activity:

a) Software requirements are documented and evaluated, reusing applicable software requirements, if any exist

6.1.2.2 Activities and tasks

6.1.2.2.1 Software requirements analysis

The following reuse-related task is added to this activity:

6.1.2.2.1.1 The developer shall include software reusability requirements for reuse assets in the quality characteristics specifications to assure the quality of the reuse assets selected for reuse in the development of the software product. The following list suggests some criteria that may be used to evaluate the quality of reusability:

a) The reliability experience of reuse assets reused in similar projects

b) The results of inspecting the reuse assets for defects

c) The results of testing the reuse assets against the system and software requirements

6.1.3 Software Architectural Design Process

6.1.3.1 Outcomes

The following reuse-related outcome is added to this activity:

a) The reusability of the software architecture is evaluated

6.1.3.2 Activities and tasks

6.1.3.2.1 Software architectural design

For each software item, the following reuse-related tasks are added to this activity:

6.1.3.2.1.1 The developer shall derive a software architecture that is based on selected, applicable domain architectures. If no such domain architectures exist, the developer may define an architecture for this software product that describes the structures of the software product and the software components that comprise this structure. The developer shall allocate software requirements to this architecture following the domain models that correspond to the selected domain architecture, if it exists. The software architecture shall be documented reusing applicable documentation reuse assets, if any exist.

6.1.3.2.1.2 Interfaces external to the software item and between the software components within the software item shall comply with the domain architecture interfaces and shall be derived from applicable reuse assets, such as interface reuse assets, if any exist.

6.1.3.2.1.3 Database designs shall be derived from selected domain models and database designs, if any exists.

6.1.3.2.1.4 User documentation shall be created reusing applicable documentation assets, if any exist.

6.1.3.2.1.5 Preliminary test requirements shall be created reusing applicable test requirements, if any exist.

6.1.3.2.1.6 The developer shall evaluate the software architecture, the interface, and database designs according to reusability criteria. The results of the evaluation shall be documented.

NOTE—Reusability criteria include, but are not limited to the following:

a) Compliance with the organization's reuse standards

b) Usability and reusability of the software architecture and database designs

c) Reuse potential of the software architecture and its components to be used in multiple contexts

6.1.3.2.1.7 If appropriate, the developer shall modify the software requirements in response to this evaluation. The developer shall re-analyze the software requirements as described in 6.1.2 of this standard.

6.1.3.2.1.8 The developer shall use a domain engineering feedback mechanism and a reuse asset management communication mechanism to report reuse information about the software architectural design.

NOTE—Reuse information about the software design includes, but is not limited to the following:

a) Problems encountered with the domain architectures or the domain models
b) The reasons for being unable to reuse the domain architectures or domain models
c) Suggested modifications to the domain architectures or domain models
d) List of domain architectures and domain models used by this project
e) List of new candidates reusable assets suggested by this project

6.1.4 Software Detailed Design Process

6.1.4.1 Outcomes

The following reuse-related outcome is added to this process:

a) A detailed design for each software item is produced reusing applicable design and documentation assets, if any exist

6.1.4.2 Activities and tasks

6.1.4.2.1 Software detailed design

For each software item, the following reuse-related tasks are added to this activity:

6.1.4.2.1.1 The developer shall select and reuse an applicable detailed design for each software component, if any exists. If none exists, the developer shall develop a new detailed design for the software component. When writing new detailed designs, the developer shall use the language and concepts from the selected domain models. The developer shall use data structures and naming conventions from the domain models when describing detailed designs.

6.1.4.2.1.2 The developer shall develop and document interfaces external to the software item, between software components, and between software units that are compliant with domain interface standards. The developer shall reuse applicable assets, if any exists.

6.1.4.2.1.3 The developer shall select and reuse applicable detailed designs and documentation for the database, if any exists.

6.1.4.2.1.4 The developer shall reuse applicable test assets to create test requirements, if any exists.

6.1.4.2.1.5 The developer shall evaluate the software detailed design and test requirements according to reusability criteria. The results of the evaluation shall be documented.

NOTE—Reusability criteria include, but are not limited to the following:

a) Consistency with the software architecture design and compliance to the domain architectures and domain models

b) Compliance to the organization's reuse standards

c) Usability and reusability of the software design and test requirements

d) Identification of customized designs and test requirements that may be replaced by reusable designs and test requirements if the software requirements are adjusted

e) Reuse potential of the software item designs, software component designs, software unit designs, and test requirements to be used in multiple contexts

6.1.4.2.1.6 The developer shall use a domain engineering feedback mechanism and a reuse asset management communication mechanism to report the reuse information about the software detailed design.

6.1.5 Software Construction Process

6.1.5.1 Activities and tasks

6.1.5.1.1 Software construction

For each software item, the following reuse-related task is added to this activity:

6.1.5.1.1.1 The implementer shall review current documentation for test results and use them for the project, if applicable. The following criterion shall be added to the results of evaluation:

a) Reusability of unit tests

6.1.6 Software integration process

6.1.6.1 Outcomes

The following reuse-related outcome is added to this process:

a) An integration plan is developed, reusing appropriate plan templates, if any exist.

6.1.6.2 Activities and tasks

6.1.6.2.1 Software integration

For each software item, the following reuse-related tasks are added to this activity:

6.1.6.2.1.1 The developer shall create and document an integration plan, reusing an applicable integration plan template, if any exists, to define the resources and procedures for integrating software units and components into the software item.

6.1.6.2.1.2 The developer shall document the software item and the test results reusing applicable documentation assets, if any exist.

6.1.6.2.1.3 The developer shall develop a set of qualification tests, test cases, and test procedures, using applicable test assets, if any exist, to test each qualification requirement of the software item.

6.1.6.2.1.4 The developer shall evaluate the integration plan, software design, code, tests, and user documentation according to reusability criteria. The results of the evaluation shall be documented.

NOTE—Reusability criteria include, but are not limited to the following:

a) Compliance to the organization's reuse standards
b) Usability and reusability of the software item, test cases, and test procedures
c) Reuse potential of the software design, code, test cases, test procedures, and user documentation to be used in multiple contexts

6.1.6.2.1.5 The developer shall use a domain engineering feedback mechanism and a reuse asset management communication mechanism to report reuse information about software integration.

6.1.7 Software Qualification Testing Process

6.1.7.1 Outcomes

The following reuse-related outcome is added to this process:

a) Software qualification testing is performed reusing applicable qualification tests, if any exist.

6.1.7.2 Activities and tasks

6.1.7.2.1 Software qualification testing

For each software item, the following reuse-related tasks are added to this activity:

6.1.7.2.1.1 The developer shall document the results of software qualification testing reusing applicable documentation assets, if any exist.

6.1.7.2.1.2 The developer shall evaluate the software design, code, test assets, and user documentation according to reusability criteria. The results of the evaluation shall be documented.

NOTE—Reusability criteria include, but are not limited to the following:

a) Compliance to the organization's reuse standards
b) Usability and reusability of the software design, code, test assets, and user documentation
c) Reuse potential of the software design, code, test assets, and user documentation to be used in multiple contexts

6.1.7.2.1.3 The developer shall use a domain engineering feedback mechanism and a reuse asset management communication mechanism to report the reuse information about the software qualification testing.

6.2 Software Support Processes

The Software Support Processes consist of the following activities:

a) Software Documentation Management Process
b) Software Configuration Management Process
c) Software Quality Assurance Process
d) Software Verification Process
e) Software Validation Process
f) Software Review Process
g) Software Audit Process
h) Software Problem Resolution Process

6.2.1 Software Documentation Management Process

6.2.1.1 Activities and tasks

6.2.1.1.1 Process implementation

The following reuse-related task is added to this activity:

6.2.1.1.1.1 Where applicable, the documents to be produced during the life cycle of the software product shall be reused from previous projects. In addition to the documentation attributes defined in IEEE Std 12207-2008, 7.2.1.3.1.1, the following shall be addressed:

a) Reusability of the software
b) Reusability of the documentation
c) Reusability of the test data

6.2.1.1.2 Design and development

The following reuse-related tasks are added to this activity:

6.2.1.1.2.1 Where applicable, the documentation standards shall apply to all documentation produced by the organization associated with the project.

6.2.2 Software Configuration Management Process

6.2.2.1 Activities and tasks

6.2.2.1.1 Process implementation

The following reuse-related tasks are added to this activity:

6.2.2.1.1.1 If applicable, an existing software configuration management plan shall be reused.

6.2.2.1.1.2 The developer shall document all reusable assets utilized throughout the project, archiving them for use in future projects.

6.2.3 Software Quality Assurance Process

6.2.3.1 Activities and tasks

6.2.3.1.1 Process implementation

The following reuse-related tasks are added to this activity:

6.2.3.1.1.1 The developer shall include software reusability requirements for assets in the quality characteristics to assure the quality of the reuse assets selected for reuse in the development of the software product. The following criteria may be used to evaluate the quality of reusability:

a) The reliability experience of reuse assets reused in similar projects

b) The results of inspecting the reuse assets for defects

6.2.3.1.1.2 The development shall document the results of software qualification testing reusing applicable documentation assets, if any exist.

6.2.3.1.1.3 The developer shall evaluate the software design, code, and user documentation according to reusability criteria. The results of the evaluation shall be documented.

6.2.4 Domain Engineering Process

6.2.4.1 Activities and tasks

6.2.4.1.1 Process implementation

The following reuse-related tasks are added to this activity:

6.2.4.1.1.1 The domain engineer shall create and document a domain engineering plan, reusing an applicable do- main engineering plan template, if any exists, to define the resources and procedures for performing domain engineering. The plan should include standards, methods, tools, activities, assignments, and responsibilities for performing domain engineering. To create the domain engineering plan, the domain engineer should consult the literature and/or data resources about the domain and should consult with domain experts, developers, and users of software products within the domain.

6.2.4.1.1.2 The domain engineer shall select the representation forms to be used for the domain models and domain architectures, in accordance with the organization’s reuse standards, and by consulting domain experts, developers, and users of software products within the domain.

6.2.5 Reuse asset management process

6.2.5.1 Outcomes

The following reuse-related outcomes are added to this process:

a) The reuse of each asset, modification requests for the asset, and problem reports concerning the asset are tracked; and

b) Each accepted asset is classified according to the reuse classification scheme and made available through an asset storage and retrieval mechanism.

Annex A

(informative)

Bibliography

[B1] IEEE Std 610.12™-1990, IEEE Standard Glossary of Software Engineering Terminology. [9, 10]

[B2] IEEE Std 1067™-2005, IEEE Guide for In-Service Use, Care, Maintenance, and Testing of Conductive Clothing for Use on Voltages up to 765 kV ac and 1750 kV dc.

[B3] IEEE Std 1517™-1999 (Reaff 2004), IEEE Standard for Information Technology—Software Life Cycle Processes—Reuse Processes.

[B4] IEEE Std 15288™-2008, IEEE Standard for Systems and software engineering—System Life Cycle Processes.

[B5] IEEE Std 24765™-2010, IEEE Standard for Systems and Software Engineering—Vocabulary.

[B6] ISO/IEC 12207:2008, Systems and software engineering—Software life cycle processes.[11]

[B7] ISO/IEC 15288:2008(E), Systems and software engineering —System life cycle processes.

[B8] ISO/IEC 24765:2009, Systems and software engineering—Vocabulary.

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

[10] The IEEE standards or products referred to in Annex A are trademarks owned by the Institute of Electrical and Electronics Engineers, Incorporated.

[11] ISO publications are available from the ISO Central Secretariat, 1, ch. de la Voie-Creuse, Case Postale 56, CH-1211, Geneva 20, Switzerland (http://www.iso.org/). IEC publications are available from the Central Office of the International Electrotechnical Commission, 3, rue de Varembé, P.O. Box 131, CH-1211, Geneva 20, Switzerland (http://www.iec.ch/). ISO/IEC 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/).

Annex B

(informative)

Basic concepts

This annex contains the concepts upon which this standard was developed. The information in this annex is intended as an aid in understanding the concepts underlying this standard.

B.1 Software processes

Software processes are a road map or framework to developing systems. Software processes should provide a frame- work for practicing reuse that lays out reuse activities. Only when reuse considerations are integrated into the software life cycle can reuse become a natural and normal way of working.

A reuse-based software life cycle describes how software products are built through the use of assets and how to build and manage assets.

In order to practice reuse in the context of the software life cycle, determinations must be made about generally how reuse changes the software life cycle, exactly where to fit reuse into the software life cycle processes, and specifically what reuse outcomes, processes, activities, and tasks should be included in the software life cycle.

B.2 Reuse changes the software life cycle

Reuse changes the software life cycle because reuse is a fundamentally different software paradigm.

Certainly, there can be no argument that reuse is a different model. The traditional approach to software development is altered by reuse because reuse forces a different way of thinking and working. Its basic operating premise is: *Do not build from the first principle; that is, build software from scratch only as a last resort.* This is directly contrary to the basic operating premise found in most conventional software life cycle models, where it is assumed, often by default, that life cycle deliverables are produced from scratch. For example, most software life cycle models do not describe how to create a design by reusing a domain architecture; nor do they describe how to build a domain architecture that can be reused in building a set of related software products.

To enable reuse, the software life cycle must start with reuse, end with reuse, and include reuse in all phases in between. A common practice has been to postpone reuse until the end of the life cycle. Treating reuse strictly as an end life cycle activity, however, limits the reuse possibilities and benefits. To maximize the benefits that reuse can deliver, it is important to “front load” the software life cycle with reuse thinking and planning.

The following three basic reuse characteristics should be incorporated into the software life cycle to promote and enable the practice of reuse:

- Reuse themes that govern software development decision making
- Reuse requirements that broaden the focus from one software product at a time to multiple, related software products over time
- Reuse views that slice the software process into two separate sides or views of reuse

B.2.1 Reuse themes

The concept of "reuse thinking" comprises the following three reuse themes that developers must constantly keep in mind when developing software products and making development decisions:

— Take advantage of existing assets

— Identify reuse opportunities

— Prepare for reuse down the process road

Software developers must consider all of these reuse themes when making every type of development decision, from planning strategies to implementation tools to language choices. A developer who employs reuse thinking is aware of what has been previously built and, whenever possible, reuses existing assets to produce new software products and enhancements to existing systems.

Reuse thinking avoids the "copy and modify" syndrome that typifies much software development. Any software asset is likely to change over time. Effective reuse of an asset, however, does not result in a succession of similar but subtly different versions, each of which must be maintained as a distinct asset.

The first reuse theme, *Take advantage of existing assets,* emphasizes the need to leverage previously completed work when possible. Custom building software products from scratch has the benefit of enabling a developer to exactly match software product requirements. By modifying existing assets, however, a developer can produce the software product in a faster, more cost-effect manner. These trade-offs must be considered in the development planning process. With reuse, software product requirements specifications are adjusted to reuse available assets unless sound business and/or technical reasons that override taking advantage of reuse opportunities can be demonstrated. Reuse thinking allows software product requirements, especially optional requirements, to be influenced by what exists and can be reused.

The second reuse theme, *Identify reuse opportunities,* requires constantly looking for opportunities to practice reuse. Every project deliverable to be produced is potentially reusable in other software products and projects. Determining a project deliverable's reuse potential is an integral part of building it. If valuable, the asset's specification, design, documentation, and so on should be created in a manner that enables its effective reuse. It may be necessary to look beyond project boundaries to determine reuse potential.

Although the second reuse theme requires a mindset concerned with one life cycle process, the third reuse theme, *Prepare for reuse down the process road,* requires consideration of other life cycle processes as well. For example, reuse requires that software design is examined with respect to not only its quality, but also how well it maximizes implementation reuse opportunities (e.g., reusing existing code or test cases, or using generator tools).

B.2.2 Reuse requirements

Reuse places a unique set of demands on the software life cycle. The requirements that reuse places on the software life cycle include the following:

— Multi-system perspective where the project focus broadens beyond the requirements of a single system or single software product to take into account common requirements shared with other systems or software product implementations now and in the future.

— References to the past to leverage legacy systems by their reuse in part or in whole, and references to the future to leverage strategic systems plans to defined a business-based scope for the reuse practice in an organization.

— Exploitation of commonality and accommodation of variability across a set of software products to identify assets that are common and reusable.

When practicing reuse, developers must not approach a software product in isolation. They must think in terms of a group or family of products linked by the requirements, features, and functions that they share. Whenever developers build assets to implement these commonalities, they attempt to build the assets once in a way that allows them to be reused wherever the requirements occur. This technique often entails a generalization of the assets, compliance to standards, and a design strategy that stresses consistency across software product implementations.

Commonality/variability analysis is the process of identifying common assets in a set of existing and future software products. Its purpose is to determine whether it is advantageous to develop, re-engineer, or acquire reusable assets that then can be used in the creation and maintenance of software products in this set. Both the similarities and differences in the common assets are examined. A reusable asset must be designed to be easily adjusted to handle possible variations in the asset that each reuse requires.

A reuse software life cycle must incorporate specific ways to handle each of the preceding requirements that reuse places on the software life cycle. Like reuse themes, reuse requirements run through and impose changes to the entire life cycle. For each software life cycle process, specific outcomes, activities, and tasks critical to achieving these reuse requirements should be added.

B.3 Reuse views

Reuse changes how software developers view the software life cycle. Reuse divides the software life cycle into the following distinct views:

— Activities for using reusable assets in the creation of new software products (i.e., development with reuse view)

— Activities for creating, acquiring, or re-engineering reusable assets (i.e., development for reuse view)

B.3.1 Development with reuse

Development with reuse requires that reuse thinking occurs in every development activity, including project planning, implementation, and testing. Every opportunity to develop the software product and its related project deliverables, such as user documentation or test plans, from assets should be explored. The project banner reads: *Create from scratch only as a last resort.* Each development decision should be made in the context of what assets are available for reuse.

Although development with reuse does not include creating new assets, all software deliverables should be created with reuse in mind (i.e., with the assumption that any deliverable may eventually be reused within or outside of the project).

In general, the following reuse activities should be integrated into a software life cycle to enable development with reuse:

— Extending the software development pan to include a software reuse plan that defines the following:

  a) What assets are available for reuse in the production of the software product

  b) Reuse target levels for the software project

  c) Tools needed to support the practice of reuse in the project

d) How to measure the impact of reuse on the project

— Searching for and evaluating application packages for use in the project

— Evaluating the benefits and costs associated with practicing reuse in the project

— Searching the reuse libraries and other internal and external sources for assets that can be reused to produce the software product

— Reusing assets possibly by deriving modified/specialized instances in the resulting project deliverables while maintaining the deliverables from the generic asset level

— Including reuse checks in project reviews, inspections, and audits

B.3.2 Development for reuse

The creation of new assets is a secondary concern of development with reuse; it is the primary focus of development for reuse. Developing for reuse entails conducting projects, such as building domain models and architectures, and building new assets, or re-engineering existing assets to improve their reusability.

The following activities should be added to the software life cycle to support development for reuse:

— Performing domain analysis

— Performing domain design

— Building reusable assets

B.4 Observations about practicing reuse

The three general observations to note about making reuse explicit within the software life cycle are as follows:

— Practicing reuse (requires a broad-based, multi-system perspective). The project teams should broaden its vision from the current project and software product needs to also consider what is being produced and what is needed by other concurrent and future projects and software products. This enables project teams to take advantage of what is available to reuse and to recognize reuse opportunities now and in the future.

— Planning is key. The sooner a reuse opportunity is recognized, the better the chance the project team has to incorporate it into the project. The emphasis should be on higher-level, early life cycle deliverables and bigger assets (e.g., domain architectures) since this approach brings bigger reuse benefits to the project.

— Good and continual communication within and across project teams is essential. Reuse, even at the project level, demands that communication links be established between the project and other projects, and between software developers and domain engineers.

Annex C

(informative)

Tools

Tools that provide automated support for reuse make reuse easier to practice and help improve the quality of assets. Reuse-oriented tools extend or complement software development tools to the extent that they handle the reuse properties of assets. Table C.1 describes only the types of tools needed to support reuse, but not specific examples of such types of tools. While some of the categories of reuse tools listed in Table C.1 have been manual activities, and may continue to be, all categories have the potential to benefit from automation.

Table C.1—Reuse support tool categories

Tool category	Type of tool	Functions performed by tool
Analysis and design	Reuse-oriented domain analysis and design	Assist domain engineers to recognize similarities among domain elements and to trial-fit elements into existing models and architectures. Assist developers to extend and improve their inventory of domain models and architectures.
	Legacy-asset salvage analysis	Analyze legacy assets in order to determine structural and functional patterns of similarity
	Applications requirements analysis	Cross-match requirements to existing assets in order to minimize the deltas between what is available and what is needed.
	Reuse-oriented application design	Interrogate selected domain architectures in order to present developers with a list of options for instantiating the architectures' components. The result is a format specification of a software product sufficient to drive both documents and construction tools.
Asset constructors	Smart editors	Find appropriate assets and parse them so that a developer can instantiate them to a particular context.
	Generators	Construct assets by combining design specifications with domain information contained within the tool.
	Assemblers	Construct assets by combining design specifications with assets external to the tool.
	Legacy-asset reconditioners	Package desired patterns, extracted by salvage analysis tools, into assets.
Asset testers	Adaptability testers	Assist domain engineers to determine and improve the ease-of-reuse of given assets.
	Generality testers	Assist domain engineers to determine and modify the domain of applicability of given assets.
Reuse management	Measuring reuse cost/benefits: Reuse asset life cycle and application life cycle	Determine costs to manage the asset storage and retrieval mechanisms and to amortize assets over time and over software products. Determine the relative costs and benefits of having various assets. Determine project development and maintenance time and effort expended/avoided due to reuse.
	Asset configuration and version management	Keep track of how to access needed assets, ownership, and servicing responsibilities, and which version of an asset applies to which software products.
	Impact analysis of asset modification	Keep track of where assets are reused, and dependencies among assets.
	Asset inventory analysis	Determine the orthogonality (duplication and/or overlap) of the inventory, and the age and status of inventory items.
	Asset cataloging	Formal registration of assets into various asset storage and retrieval mechanisms, including updating browsing and retrieval tools with appropriate descriptors and search criteria.
	Asset search and retrieval	Browse and access assets, possibly allowing developers to enter appropriate parameter settings for both construction-time adaptation and run-time execution.
	Asset certification	Support the secure certification of an asset’s status in terms of its scope of reusability—project, department, enterprise, industry, etc.

Annex D

(informative)

Glossary

This annex contains definitions for software reuse terms that are of general interest but not specifically referenced in this standard.

assembly: The part of compileable (or interpretable) software unit that results from assembling other parts.

asset pattern: An asset that formally describes the characteristics of a set of assets.

black-box reuse: The reuse of unmodified software components.

consumer: A person or organization that reuses software.

context analysis: The process of determining the scope and boundary of a domain.

domain implementation: The process of creating adaptable assets that can be reused in the development of software systems within a domain. Domain implementation may also include the specification of a software development process that describes how software systems in the domain are developed through reuse of assets.

domain-specific language: A problem- or task-oriented modeling language used to simplify the process of assembling, customizing, generating, or configuring a system or component.

domain-specific reuse: The reuse of assets with an application domain.

frame: A parameterized part that allows instantiation by other frames.

framework: A set of cooperating classes or frames that makes up a reusable design for a specific class of software.

generator: A mechanism that creates assets by inclusion of information that is internal to the generator.

gray-box reuse: Reuse of a part by applying a few changes to the part.

horizontal reuse: A type of reuse where assets are used across domains.

life cycle reuse percentage: The reuse level of a project measured by summing the actual levels of reuse during each phase of the software life cycle and adjusting those levels for the amount of effort expended in that life cycle phase.

opportunistic reuse: The practice of reuse in an informal way without taking place in a global reuse improvement strategy.

organized reuse: The practice of reuse according to a long-term plan.

part: An asset that can be assembled into a compileable or interpretable software unit.

planned reuse: *See:* **organized reuse.**

relative cost of reuse: The portion of the effort that it takes to reuse a component without modification versus writing it anew.

relative cost of writing for reuse: The portion of the effort that it takes to write a reusable component versus writing it for one-time use only.

reusable component: *See:* **part.**

reuse catalog: A set of descriptions of assets with a reference or pointer to where the assets are actually stored or information about how they can be acquired.

reuse interface: A set of parameters that governs the adaptation of assets.

reuse interoperability: An indication of the capability to exchange assets, asset descriptions, and other information among reuse libraries.

reuse library: A classified collection of assets that allows searching, browsing, and extracting.

reuse maturity: A variation of the Software Engineering Institute (SEI) Process Capability Maturity Model (CMM) used to describe how well an organization has developed and practices reuse.

reuse ratio: The proportion of software in a given software product that came from reusable parts.

salvage: The process of finding and re-engineering an existing asset so that it may potentially be reused in subsequent software products, developments, or maintenance.

scalability: The degree to which assets can be adapted to support application engineering products for various defined measures.

vertical reuse: A type of reuse in which assets are reused within an application domain.

white-box reuse: Reuse of a component by applying major changes to the component.