

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

Sponsor

**Software Engineering Standards Committee
of the
IEEE Computer Society**

Reaffirmed 25 March 2004

Approved 26 June 1999

IEEE-SA Standards Board

Abstract: A common framework for extending the software life cycle processes of IEEE/EIA Std 12207.0-1996 to include the systematic practice of software reuse is provided. This standard specifies the processes, activities, and tasks to be applied during each phase of the software life cycle to enable a software product to be constructed from reusable assets. It also specifies the processes, activities, and tasks to enable the identification, construction, maintenance, and management of assets supplied.

Keywords: asset, domain engineering, process, software life cycle, software reuse

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

Copyright © 1999 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 14 July 1999. Printed in the United States of America.

Print: ISBN 0-7381-1735-8 SH94751
PDF: ISBN 0-7381-1736-6 SS94751

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

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

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.

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 all 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.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

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

Authorization to photocopy portions of any individual standard for internal or personal use is granted by the Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; (978) 750-8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Introduction

(This introduction is not part of IEEE Std 1517-1999, IEEE Standard for Information Technology—Software Life Cycle Processes—Reuse Processes.)

Implementing software reuse entails more than creating and using libraries of assets. It requires formalizing the practice of reuse by integrating reuse processes and activities into the software life cycle. Unless reuse is explicitly defined in the software life cycle processes, an organization will not be able to repeatedly exploit reuse opportunities in multiple software projects or products. Without this repetition, the improvement to the software life cycle and the software products that result from the practice of reuse will be limited and, perhaps, disappointing.

Systematic reuse is the practice of reuse according to a well-defined, repeatable process. Practicing systematic reuse enables significant software productivity, quality, and cost improvements. The major benefits that systematic reuse can deliver are as follows:

- Increase software productivity;
- Shorten software development time;
- Move personnel, tools, and methods more easily from project to project;
- Reduce software development and maintenance costs;
- Produce higher quality software products;
- Improve software product interoperability;
- Provide a competitive advantage to an organization that practices reuse.

There are a variety of approaches to implement the concept of reuse. What distinguishes systematic reuse is the avoidance of the proliferation of multiple versions of otherwise common elements. For example, if a reuse approach results in multiple instantiations of a common element, each of which may be modified by software developers, then the element is no longer common and can no longer be maintained as a single element apart from its instantiations. In the context of this standard, systematic reuse excludes those approaches.

The potential for reuse is enormous since the majority of each software product can be built with assets, assuming they are available. An asset is any item, such as a design or test plan, that has been designed to be used in multiple contexts, such as multiple software products, multiple implementations of a software product, or multiple software projects.

One major problem encountered by organizations attempting to practice reuse is that reuse is simply missing from their software life cycle processes. The intent of this standard is to address this problem by defining a common framework for reuse processes and by defining how to integrate the practice of reuse into traditional software life cycle processes.

Reuse processes describe how software products are built with assets and how to build and manage these assets. The reuse process framework presented in this standard covers both the life cycle of a software product that is developed from assets and the life cycle of an asset.

IEEE/EIA Std 12207.0-1996 defines software life cycle processes in general. This standard adds to IEEE/EIA Std 12207.0-1996 the life cycle processes and activities that enable the practice of systematic reuse. Thus, the use of this standard requires access to and an understanding of IEEE/EIA Std 12207.0-1996.

For organizations that already employ systematic reuse processes, this standard may be used to determine the conformity of those processes to this standard, and as a basis for improvement of those processes where warranted. When establishing or improving systematic reuse processes, organizations are encouraged to assess the business case. While systematic reuse enables major benefits such as those already described,

there are costs, risks, and other considerations that may prevent those benefits from being realized. These include the following:

- *The degree to which reuse benefits are relevant to the organization.* For example, a very small organization, or one that experiences insignificant time and costs to develop 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.* While the organization may wish to undertake systematic reuse, its capability maturity may be insufficient to implement the processes in this standard. Or, 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. Changing attitudes and associated non-reuse behaviors can be difficult. Policy changes and capital investments, which require senior management to be firmly committed to the achievement of systematic reuse, may be necessary.

Organizations interested in undertaking systematic 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 timeline for achieving the transition to systematic reuse, is an excellent way to ensure success.

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

IEEE Std 1517-1999 may be used to

- Acquire, supply, develop, manage, and maintain assets;
- Acquire, supply, develop, operate, and maintain software products that are built in whole or in part with assets;
- Manage and improve the organization's software life cycle processes with respect to the practice of software reuse;
- Establish software management and engineering environments based on reuse processes;
- Foster improved understanding between customers and vendors and parties involved in the reuse-based life cycle of a software product or system and assets;
- Facilitate the use of assets to develop software products and systems;
- Facilitate the development of assets.

IEEE Std 1517-1999 has been written to work with and integrate into IEEE/EIA Std 12207.0-1996.

Participants

This standard was prepared by the Working Group on Information Technology—Software Life Cycle Processes—Reuse Processes. The work was coordinated with the Reuse Steering Committee (RSC) of the IEEE Computer Society Software Engineering Standards Committee (SESC).

At the time this standard was completed, the Reuse Processes Working Group had the following membership:

Carma McClure, Chair

Rose Armstrong
Paul Bassett
Steve Cichinski
Verlynda Dobbs
Patrick Donohoe
Simon Heesh

Kay Hohn
Frank Huy
Rebecca Joos
Tony Kram
Ann Laign
Anil Midha

Nader Nada
Tim Niesen
Juan Roa
Mia Watts
Jim Wilson
Kathryn P. Yglesias

The following members of the balloting committee voted on this standard:

Jeremy A. Adams
Bakul Banerjee
Edward E. Bartlett
Barbara K. Beauchamp
Leo Beltracchi
Richard E. Biehl
Juris Borzovs
James E. Cardow
Enrico A. Carrara
Lawrence Catchpole
Keith Chan
Keith Chow
Antonio M. Cicu
Paul R. Croll
Geoffrey Darnton
Taz Daughtrey
Bostjan K. Derganc
Verlynda Dobbs
George Duffield
Leo G. Egan
William Eventoff
Richard E. Fairley
John W. Fendrich
Neal Fishman
Jay Forster
John H. Fowler
John Garth Glynn
Julio Gonzalez-Sanz
L. M. Gunther
David A. Gustafson

John Harauz
William Hefley
Debra Herrmann
John W. Horch
George Jackelen
Frank V. Jorgensen
Vladan V. Jovanovic
William S. Junk
Robert J. Kierzyk
Thomas M. Kurihara
John B. Lane
Dieter Look
Stan Magee
Henry A. Malec
Tomoo Matsubara
Jacques Meekel
Denis C. Meredith
Jair Merlo
James Bret Michael
Kathryn J. Moland
James W. Moore
Francisco Navas Plano
Susumu Ohno
Gerald L. Ourada
A. Pedar
Donald J. Pfeiffer
Alex Polack
Jeffrey S. Poulin
Kenneth R. Ptack

Surendra K. Reddy
Donald J. Reifer
Annette D. Reilly
Dennis Rilling
Andrew P. Sage
Helmut Sandmayr
Frederico Sousa Santos
Robert J. Schaaf
Norman Schneidewind
David J. Schultz
Carl A. Singer
Mitchell W. Smith
JagSodhi
Alfred R. Sorkowitz
Mark Spillers
Luca Spotorno
Fred J. Strauss
Toru Takeshita
Richard H. Thayer
Booker Thomas
Leonard L. Tripp
Delores Wallace
Scott A. Whitmire
Paul A. Wolfgang
Paul R. Work
Kathryn P. Yglesias
Natalie C. Yopconka
Janusz Zalewski
Geraldine Zimmerman
Peter F. Zoll

When the IEEE-SA Standards Board approved this standard on 26 June 1999, it had the following membership:

Richard J. Holleman, *Chair*

Donald N. Heirman, *Vice Chair*

Judith Gorman, *Secretary*

Satish K. Aggarwal
Dennis Bodson
Mark D. Bowman
James T. Carlo
Gary R. Engmann
Harold E. Epstein
Jay Forster*
Ruben D. Garzon

James H. Gurney
Lowell G. Johnson
Robert J. Kennelly
E. G. "Al" Kiener
Joseph L. Koepfinger*
L. Bruce McClung
Daleep C. Mohla
Robert F. Munzner

Louis-François Pau
Ronald C. Petersen
Gerald H. Peterson
John B. Posey
Gary S. Robinson
Akio Tojo
Hans E. Weinrich
Donald W. Zipse

*Member Emeritus

Also included is the following nonvoting IEEE-SA Standards Board liaison:

Robert E. Hebner

Catherine K.N. Berger
IEEE Standards Project Editor

Contents

1. Overview	1
1.1 Scope.....	1
1.2 Purpose.....	2
1.3 Field of application	2
1.4 Conformance.....	2
2. References.....	3
3. Definitions.....	3
4. Application of this standard	5
4.1 Organization of this standard	5
5. Integration of reuse into the primary life cycle processes	7
5.1 Acquisition Process.....	7
5.2 Supply Process.....	9
5.3 Development Process.....	10
5.4 Operation Process	17
5.5 Maintenance Process.....	18
6. Reuse supporting process.....	20
6.1 Asset Management Process.....	20
7. Reuse organizational life cycle process	22
7.1 Reuse Program Administration Process.....	22
8. Reuse cross-project life cycle process	26
8.1 Domain Engineering Process.....	26
Annex A (informative) Bibliography	30
Annex B (informative) Basic concepts	31
Annex C (informative) Reuse process objectives	35
Annex D (informative) Relationships	39
Annex E (informative) Tools	40
Annex F (informative) Glossary	42

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

1. Overview

This standard provides a common framework for extending the software life cycle processes to include the systematic practice of software reuse. It specifies the processes, activities, and tasks to be applied during each phase of a software life cycle to enable a software product to be constructed from assets.

This standard also specifies the processes, activities, and tasks to enable the identification, construction, maintenance, and management of assets.

1.1 Scope

This standard is intended to describe reuse processes and to relate them to the software life cycle processes that are defined in IEEE/EIA Std 12207.0-1996.¹ (See Annex D of this document.) In this sense, this standard supplements IEEE/EIA Std 12207.0-1996. This standard uses process specifications from IEEE/EIA Std 12207.0-1996 to describe reuse processes.

The scope of this standard will be broader than a single project because reuse processes, such as Domain Engineering, transcend the life cycle of any single software product and apply to a set of related software products.

This standard describes reuse processes at the reference level, as defined by the Basili architecture for process abstraction [B1].² In the Basili architecture, the responsibility for each process is assigned to a party referred to as the “agent.” From this viewpoint, this standard can be viewed as a list of agents and their minimum responsibilities.

Like IEEE/EIA Std 12207.0-1996, this standard uses the term *party* to indicate the individual or organization responsible for executing a process or activity. The party is often designated with a name derived from the name of the process or activity. For example, the party (individual or group) responsible for Domain Engineering may be termed the *domain engineer*.

¹Information on references can be found in Clause 2.

²The numbers in brackets correspond to those of the bibliography in Annex A.

The limitations of this standard are in

- Describing a high-level framework for reuse processes, but not the details of how to perform the activities and tasks included in the processes;
- Describing the responsibilities inherent in the various processes, but not the detailed data or control connections among the processes;
- Specifying software life cycle reuse processes at the reference level, but not prescribing a specific software life cycle process model or software methodology;
- Viewing reuse processes as an assignment of continuing responsibilities to agents, but not as a series of steps (procedures) to be performed;
- Specifying provisions for acquiring 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/EIA Std 12207.0-1996, there are a number of lists for tasks; none of these is presumed to be exhaustive—they are intended as examples.

1.2 Purpose

The purposes of this standard are to

- Establish a framework for practicing reuse within the software life cycle;
- Define the processes, activities, and tasks that are needed to perform and manage the practice of reuse in a single software project, across multiple software projects, and across an organization;
- Facilitate communication between software acquirers, suppliers, developers, reuse program administrators, asset managers, and domain engineers by providing a common understanding of reuse and by standardizing reuse terminology;
- Integrate reuse processes, activities, and tasks with the software life cycle processes described in IEEE/EIA Std 12207.0-1996.

1.3 Field of application

This standard applies to the acquisition of software products, including assets; to the acquisition of software services; and to the supply, development, operation, and maintenance of software products, including assets. Off-the-shelf software that is designed for reuse is a special case of assets that is covered by this standard.

The reuse processes 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 either internal or external to the organization.

1.4 Conformance

This standard adds activities and tasks to the existing processes of IEEE/EIA Std 12207.0-1996. In addition, this standard specifies processes in addition to those of IEEE/EIA Std 12207.0-1996. Conformance to this standard requires conformance to IEEE/EIA Std 12207.0-1996. (Note that IEEE/EIA Std 12207.0-1996 uses the term *compliance* rather than *conformance*; for the purpose of this standard, the two terms are considered to be synonymous.)

An organization may claim “selective conformance” to a designated process of this standard 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 a process of this standard is specified in IEEE/EIA Std 12207.0-1996, then conformance requires implementation of the corresponding requirements of IEEE/EIA Std 12207.0-1996 as well as the requirements of this standard.

An organization may claim “full conformance” to this standard if it is in selective conformance with every process specified in this standard.

2. References

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

The following standards contain provisions, which through reference in this standard, constitute provisions of this standard. At the time of publication, the standard editions indicated were the most recent.

IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology.³

IEEE/EIA Std 12207.0-1996, IEEE/EIA Standard for Information Technology—Software life cycle processes.

IEEE/EIA Std 12207.1-1997, IEEE/EIA Guide for Information Technology—Software life cycle processes—Life cycle data.

3. Definitions

For the purposes of this standard, the following terms and definitions apply. IEEE Std 100-1996, IEEE/EIA Std 12207.0-1996, and IEEE Std 610.12-1990 should be referenced for terms not defined in this clause.

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

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

3.3 asset: An item, such as design, specifications, 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.

3.4 classification: The manner in which the assets are organized for ease of search and extraction within a reuse library [B4].

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

3.6 domain: A problem space [B7].

3.7 domain analysis: (A) The analysis of systems within a domain to discover commonalities and differences among them. (B) The process by which information used in developing software systems is identified, captured, and organized so that it can be reused to create new systems, within a domain. (C) The result of the process in (A) and (B) [B7].

³IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA (<http://www.standards.ieee.org/>).

3.8 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 [B7]. 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.

3.9 domain definition: The process of determining the scope and boundaries of a domain. The result of the process [B7].

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

3.11 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 [B7].

3.12 domain expert: An individual who is intimately familiar with the domain and can provide detailed information to the domain engineers. [B7].

3.13 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 [B7].

3.14 product line: A collection of systems that are potentially derivable from a single domain architecture.

3.15 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.

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

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

3.18 system: An integrated composite that consists of one or more of the processes, hardware, software, facilities, and people, that provides a capability to satisfy a stated need or objective [IEEE/EIA Std 12207.0-1996].

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

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

3.21 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.

4. Application of this standard

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

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

4.1 Organization of this standard

4.1.1 Reuse processes, activities, and tasks

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

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

This standard uses the software life cycle framework that is described in IEEE/EIA Std 12207.0-1996 as the foundation for describing a software life cycle that includes the practice of reuse. Reuse processes, activities, and tasks are integrated into the IEEE/EIA Std 12207.0-1996 software life cycle framework, as depicted in Figure 1.

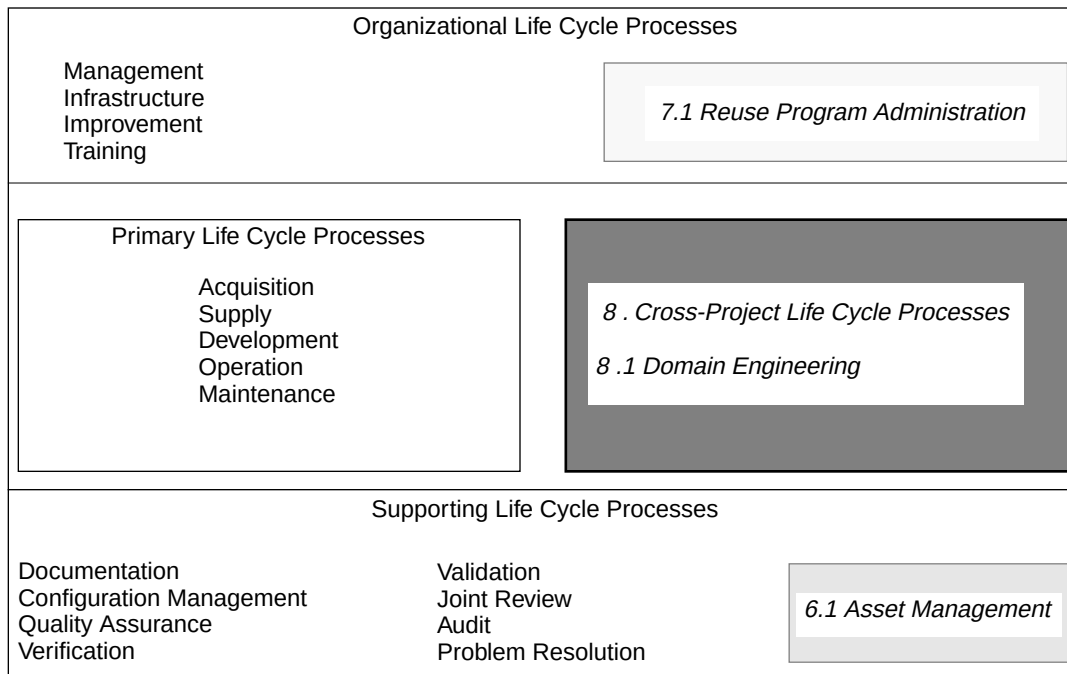


**Figure 1—Structure of IEEE Std 1517-1999 within the 12207 framework;
and framework for reuse life cycle processes**

Each life cycle process discussed in this standard is divided into a set of activities; and the requirements of each activity are specified in a set of tasks. Subclause number $x.x$ denotes a process, $x.x.x$ denotes an activity, and $x.x.x.x$ denotes a task. Correlation of the processes, activities, and tasks in this standard with those in IEEE/EIA Std 12207.0-1996 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/EIA Std 12207.0-1996. When a task defined in this standard adds requirements to an existing task in IEEE/EIA Std 12207.0-1996, the task number of the affected IEEE/EIA Std 12207.0-1996 task is indicated in the text for the corresponding reuse task specified in this standard.

4.1.1.1 Development of software products with assets

Reuse activities and tasks that are related to the development, operation, and maintenance of software products with assets are integrated into the IEEE/EIA Std 12207.0-1996 primary life cycle processes (Clause 5). These primary life cycle processes are employed to acquire, supply, develop, operate, and maintain software products. With the addition of these reuse activities and tasks, the primary life cycle processes described in this standard employ domain models, domain architectures, and assets to develop and maintain software products.

The acquirer, supplier, developer, operator, and maintainer are the parties or agents responsible for performing the primary processes.

4.1.1.2 Development of assets

This standard groups the activities and tasks that fall into the second reuse category (i.e., development and maintenance of assets) into the Domain Engineering Process (Clause 8). This reuse process is specified as a cross-project life cycle process. As the name suggests, this type of process operates across multiple projects. The Domain Engineering Process operates across projects by providing assets that can be used by multiple projects. These assets can be used by the primary life cycle processes to acquire, supply, develop, operate, and maintain software products.

The domain engineer is the party or agent responsible for performing the Domain Engineering Process.

4.1.1.3 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. IEEE/EIA Std 12207.0-1996 defines an organizational life cycle process as 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.

This standard adds the Reuse Program Administration Process (Clause 7) to the set of organizational processes defined in IEEE/EIA Std 12207.0-1996. In addition, this standard employs the IEEE/EIA Std 12207.0-1996 organizational processes, such as Management, Infrastructure, Improvement, and Training, to establish an integrated reuse program structure of processes and personnel that can be managed and improved.

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

4.1.1.4 Management of assets

This standard groups the activities and tasks that fall into the fourth reuse category (i.e., management of assets) into the Asset Management Process. This process is specified as a supporting process. IEEE/EIA Std 12207.0-1996 defines a supporting process as a process that “supports another process as an integral part with a distinct purpose and contributes to the success and quality of the software project.” Asset Management supports other life cycle processes, such as Development, Maintenance, and Domain Engineering, by specifying the activities needed to manage asset certification, classification, storage, retrieval, version control, and change control.

This standard adds the Asset Management Process (Clause 6) to the set of supporting processes defined in IEEE/EIA Std 12207.0-1996. In addition, this standard employs the IEEE/EIA Std 12207.0-1996 supporting processes, such as Documentation, Configuration Management, and Joint Review, to enable the execution of reuse processes such as Domain Engineering and Reuse Program Administration.

The asset manager is the party or agent responsible for performing the Asset Management Process.

5. Integration of reuse into the primary life cycle processes

This clause defines the reuse activities and tasks required to develop, operate, and maintain software products with assets. These reuse activities and tasks are integrated into the following IEEE/EIA Std 12207.0-1996 primary processes as described in this clause:

- Acquisition Process
- Supply Process
- Development Process
- Operation Process
- Maintenance Process

The activities and tasks in each primary process are the responsibility of the party initiating and performing this process. This party assures that the process is in existence and is functional.

5.1 Acquisition Process

The Acquisition Process contains the activities and tasks of the acquirer. The Acquisition Process is employed when a party has a need to acquire a system, software product, or asset. In this situation, the party is referred to as the acquirer.

The purposes of performing the Acquisition Process are as follows:

- Define the need to acquire a system, software product, or asset;
- Prepare and issue a request for proposal (RFP);
- Select a supplier;
- Manage the Acquisition Process.

When employing a reuse-based software life cycle to develop a software product with reusable assets, the acquirer shall perform the Acquisition Process according to the activities and tasks that have been defined in the IEEE/EIA Std 12207.0-1996 Acquisition Process.

The Acquisition Process consists of the following activities:

- Initiation;
- RFP preparation;
- Contract preparation and update;
- Supplier monitoring;
- Acceptance and completion.

In addition, to assure that the Acquisition Process supports the reuse-based life cycle, the acquirer shall perform the following additional reuse-related tasks when performing these IEEE/EIA Std 12207.0-1996 Acquisition Process activities:

- Initiation;
- RFP preparation;
- Acceptance and completion.

5.1.1 Initiation

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

5.1.1.1 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 asset manager to identify assets for consideration.

5.1.1.2 The acquirer shall require assets to

- a) Be reusable;
- b) Have interfaces compatible with the domain architecture;
- c) Be compatible with the domain model.

5.1.1.3 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 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/EIA Std 12207.0-1996, task 5.1.1.6.)

5.1.1.4 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/EIA Std 12207.0-1996, task 5.1.1.7.)

5.1.1.5 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 1062, 1998 Edition [B3]. (This task defines reuse-related requirements in addition to those requirements specified in IEEE/EIA Std 12207.0-1996, task 5.1.1.8.)

5.1.2 RFP preparation

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

5.1.2.1 The acquisition documentation shall include any requirements for practicing reuse that the supplier will be required to meet. (This task defines reuse-related requirements in addition to those requirements specified in IEEE/EIA Std 12207.0-1996, task 5.1.2.1.)