Consolidated product

Software Process Assessment – Part 2 : A model for process management Version 1.00

(Formerly BPG 1.01)

PREAMBLE

In January 1993 a program of work was approved by ISO/IEC JTC1 for the development of an international standard for software process assessment. In June 1993 the SPICE Project Organisation was established with a mandate from JTC1/SC7 to:

- assist the standardisation project in its preparatory stage by developing initial working drafts;
- undertake user trials in order to gain early experience data which will form the basis for revision of the Technical Report prior to publication as a full International Standard;
- create market awareness and take-up of the evolving standard.

The SPICE Project Organisation completed its task of producing the set of working drafts in June 1995. The SPICE user trials commenced in January 1995. The working drafts have now been handed over to JTC1/SC7 for the normal process of standards development, commencing in July 1995.

So far as can be determined, intellectual property rights for these documents reside with the individuals and organisations that contributed to their development. In agreeing to take part in the Project, participants agreed to abide by decisions of the Management Board in relation to the conduct of the Project. It is in accordance with this understanding that the Management Board has now agreed to release the baseline set of documents. This introductory statement sets out the terms and conditions under which this release is permitted.

The documents as released are available freely from the SPICE Project File Server, sisyphus.cit.gu.edu.au, by anonymous ftp, or from approved mirrors of the server. A hypertext version of the documents is also available on the World Wide Web at URL http://www-sqi.cit.gu.edu.au/spice/

TERMS AND CONDITIONS

These terms and conditions apply to the set of documents developed by the SPICE Project, and published within the Project as Version 1.0, with the following titles:

– Part 1 : *Concepts and introductory guide*
– Part 2 : *A model for process management*
– Part 3 : *Rating processes*
– Part 4 : *Guide to conducting assessment*
– Part 5 : *Construction, selection and use of assessment instruments and tools*
– Part 6 : *Qualification and training of assessors*
– Part 7 : *Guide for use in process improvement*
– Part 8 : *Guide for use in determining supplier process capability*
– Part 9 : *Vocabulary*

1. You may copy and distribute verbatim copies of any or all of the Documents as you receive them, in any medium, provided that you conspicuously and appropriately publish with each copy a copy of these Terms and Conditions. You may charge a fee for the physical act of transferring a copy.
2. You may copy extracts from these documents in materials for internal or public use, providing you provide clear acknowledgment of the source of the material, by citation or other appropriate means.
3. You may not copy, modify, sub-license, or distribute the Documents except as expressly provided under these Terms and Conditions.

Released on the Authority of the SPICE Management Board:

Project Manager	Alec Dorling
Technical Centre Managers:	
Europe	Harry Barker
Canada, Central and South America	Jean-Normand Drouin
USA	Mark Paulk / Mike Konrad / Dave Kitson
Asia Pacific	Terry Rout
Members:	Catriona Mackie, Bob Smith, Emmanuel Lazinier, Jerome Pesant, Bob Rand, Arnoldo Diaz, Yossi Winograd, Mary Campbell, Carrie Buchman, Ali Azimi, Bruce Hodgen, Katsumi Shintani

Product Managers:

– Part 1 : *Concepts and introductory guide*

Product Manager: Terry Rout

– Part 2 : *A model for process management*

Product Managers: Al Graydon, Mark Paulk

– Part 3 : *Rating processes*

Product Manager: Harry Barker

– Part 4 : *Guide to conducting assessment*

Product Manager: Harry Barker

– Part 5 : *Construction, selection and use of assessment instruments and tools*

Product Managers: Mary Campbell, Peter Hitchcock, Arnoldo Diaz

– Part 6 : *Qualification and training of assessors*

Product Manager: Ron Meegoda

– Part 7 : *Guide for use in process improvement*

Product Managers: Adriana Bicego, Pasi Kuvaja

– Part 8 : *Guide for use in determining supplier process capability*

Product Manager: John Hamilton

– Part 9 : *Vocabulary*

Product Manager: Terry Rout

Acknowledgment:

Acknowledgment is made to all contributors of the SPICE project without whom the project could not have been conceived and carried through successfully.

Note on document formatting

Use the following margins for equivalent printing on A4 or US letter paper (these are NOT the SPICE standards).

Paper size	**A4**	**US letter (imperial)**
Top margin	34.1 mm or 1.34 inches	25.4 mm or 1.0 inches
Bottom margin	34.1 mm or 1.34 inches	25.4 mm or 1.0 inches
Left margin	25.4 mm or 1.0 inches	28.4 mm or 1.12 inches
Right margin	25.4 mm or 1.0 inches	28.4 mm or 1.12 inches

SOFTWARE PROCESS ASSESSMENT Part 2

A model for process management

Contents

Foreword

In June 1991, the fourth plenary meeting of ISO/IEC JTC1/SC7 approved a study period (resolution 144) to investigate the needs and requirements for a standard for software process assessment.

The results, which are documented in a Study Report (JTC1/SC7 N944R, 11 June 1992), came to the following major conclusions:

- there is international consensus on the needs and requirements for a standard for process assessment;
- there is international consensus on the need for a rapid route to development and trialing to provide usable output in an acceptable timescale and to ensure the standard fully meets the needs of its users;
- there is international commitment to resource the project with an international project team staffed by full time resource, with development being coordinated through four technical development centres in Europe, N America (2) and Asia Pacific;
- the standard should initially be published as a Technical Report Type 2 to enable the developing standard to stabilise during the period of the user trials, prior to its issuing as a full International Standard.

The new work item was approved in January 1993 by JTC1. In June 1993 the SPICE Project Organisation was established with a mandate from JTC1/SC7 to:

- assist the standardisation project in its preparatory stage to develop initial working drafts;
- undertake user trials in order to gain early experience data which will form the basis for revision of the published Technical Report prior to review as a full International Standard;
- create market awareness and take-up of the evolving standard.

The SPICE Project Organisation completed its task of producing the set of working drafts in June 1995. These working drafts have formed the basis for this Technical Report Type 2. The period of SPICE user trials commenced in January 1995 and is synchronised in phases to allow feedback to the stages of the technical work.

ISO/IEC Directives state that a Technical Report Type 2 may be used to publish a prospective standard for provisional application so that information and experience of its practical use may be gathered.

This Technical Report Type 2 consists of the following parts, under the general title *Software Process Assessment:*

- Part 1 : *Concepts and introductory guide*
- Part 2 : *A model for process management*
- Part 3 : *Rating processes*
- Part 4 : *Guide to conducting assessment*
- Part 5 : *Construction, selection and use of assessment instruments and tools*
- Part 6 : *Qualification and training of assessors*
- Part 7 : *Guide for use in process improvement*
- Part 8 : *Guide for use in determining supplier process capability*
- Part 9 : *Vocabulary*

In this part of the International Standard (Part 2) Annex A is normative and Annexes B to H are informative.

Introduction

The purpose of this part of the International Standard is to document the set of practices fundamental to good software engineering, which can be used by the other parts of this International Standard. The model contained in this document describes processes that an organization may perform to acquire, supply, develop, operate, evolve and support software and the generic practices that characterize the capability of those processes.

A set of practices forms the lowest level of the architecture. The architecture organizes the practices into a number of categories using two different approaches (see Annex C for details). The architecture distinguishes between:

– base practices which are the essential activities of a specific process, grouped into processes and process categories by the type of activity they address;

– generic practices, applicable to any process, which represent the activities necessary to manage a process and improve its capability to perform.

The model categorizes processes into five process categories.

The **Customer-Supplier** process category consists of processes that directly impact the customer, support development and transition of the software to the customer, and provide for its correct operation and use.

The **Engineering** process category consists of processes that directly specify, implement, or maintain a system and software product and its user documentation.

The **Project** process category consists of processes which establish the project, and co-ordinate and manage its resources to produce a product or provide a service which satisfies the customer.

The **Support** process category consists of processes which enable and support the performance of the other processes on a project.

The **Organization** process category consists of processes which establish the business goals of the organization and develop process, product, and resource assets which will help the organization achieve its business goals.

Each process in the model is described in terms of base practices, which are its unique software engineering or management activities. Process categories, processes, and base practices provide a grouping by type of activity. These processes and activities characterize performance of a process, even if that performance is not systematic. Performance of the base practices may be ad hoc, unpredictable, inconsistent, poorly planned, and/or result in poor quality products, but those work products are at least marginally usable in achieving the purpose of the process. Implementing only the base practices of a process may be of minimal value and represents only the first step in building process capability, but the base practices represent the unique, functional activities of the process when instantiated in a particular environment.

Evolving process capability is expressed in terms of capability levels, common features, and generic practices. A capability level is a set of common features (sets of activities) that work together to provide a major enhancement in the capability to perform a process. Each level provides a major enhancement in capability to that provided by its predecessors in the performance of a process. They constitute a rational way of progressing through the generic practices.

Capability levels provide two benefits: they acknowledge dependencies among the practices of a process, and they help an organization identify which improvements it might perform first, based on a plausible sequence of process implementation. There are six capability levels in the model.

Level 0; *Not-Performed:* There is general failure to perform the base practices in the process. There are no easily identifiable work products or outputs of the process.

Level 1; *Performed-Informally:* Base practices of the process are generally performed. The performance of these base practices may not be rigorously planned and tracked. Performance depends on individual knowledge and effort. Work products of the process testify to the performance. Individuals within the organization recognize that an action should be performed, and there is general agreement that this action is performed as and when required. There are identifiable work products for the process.

Level 2; *Planned-and-Tracked:* Performance of the base practices in the process is planned and tracked. Performance according to specified procedures is verified. Work products conform to specified standards and requirements.

The primary distinction from the Performed-Informally Level is that the performance of the process is planned and managed and progressing towards a well-defined process

Level 3; *Well-Defined:* Base practices are performed according to a well-defined process using approved, tailored versions of standard, documented processes.

The primary distinction from the Planned-and-Tracked Level is that the process of the Well-Defined Level is planned and managed using an organization-wide standard process.

Level 4; *Quantitatively-Controlled:* Detailed measures of performance are collected and analyzed. This leads to a quantitative understanding of process capability and an improved ability to predict performance. Performance is objectively managed. The quality of work products is quantitatively known.

The primary distinction from the Well-Defined Level is that the defined process is quantitatively understood and controlled.

Level 5; *Continuously-Improving:* Quantitative process effectiveness and efficiency goals (targets) for performance are established, based on the business goals of the organization. Continuous process improvement against these goals is enabled by quantitative feedback from performing the defined processes and from piloting innovative ideas and technologies.

The primary distinction from the Quantitatively-Controlled Level is that the defined process and the standard process undergo continuous refinement and improvement, based on a quantitative understanding of the impact of changes to these processes.

A common feature in the model is a set of generic practices that address the same aspect of process implementation or institutionalization. A generic practice is an implementation or institutionalization practice (activity) that enhances the capability to perform any process. The generic practices characterize good process management that results in an increasing process capability for any process. A planned, well-defined, measured, and continuously improving process is consistently performed as the generic practices are implemented for a process. This process capability is built on the foundation of the base practices that describe the unique, functional activities of the process.

Table 1 shows the main audiences for this part of the International Standard, why each group needs the model, and how and when it will be used.

Table 1. Use of this model.

Who	Why	How	When
Software organization	Understand what to do to improve software processes	As a working guide to software project management on processes and practices to implement	During the implementation of the organization's software processes
		As a reference guide to highlight process and practices considerations	During the development/review of the organization's software processes and as a part of continuous improvement activities.
		As a training document	Same as above
	Understand which processes and practices an assessor may evaluate	As a practice checklist	Prior to an assessment
Software process assessors	Determine how an organization manages software processes and their results	As a practice checklist	Prior to and during a software process assessment

Within this part of the International Standard:

- clause 4 defines the nomenclature used to identify processes and practices and classify them within the architecture of this model;
- clause 5 defines the capability levels, common features, and generic practices that describe the capability of the processes listed in clause 6;
- clause 6 categorizes processes into five process categories and describes each process in terms of its unique base practices;
- annex A (normative) contains the requirements for creating an extended process when tailoring this part of the International Standard to the unique needs of an industry sector or organization;
- annex B provides a helpful summary list of the generic practices and base practices within the model;
- annex C describes the architecture of the model graphically and in detail.
- annexes D, E, F, and G provide detailed guidance on understanding the intent of the processes and practices in this model by, respectively:
  - providing guidance on how to interpret the generic practices when applied to a specific process;
  - mapping the processes in this model to ISO 12207;
  - mapping the processes in this model to ISO 9001; and
  - listing the sources for the processes and practices referenced in this document;
- annex H contains a style guide for constructing extended processes.

1 Scope

This part of the International Standard defines a model for software processes and practices that forms the basis for software process assessment. The model defines, at a high level, the fundamental activities (practices) that are essential to good software engineering. It describes what activities are required, not how they are to be implemented. This model for processes and process management is applicable to all software organizations and does not presume particular organizational structures, management philosophies, software life cycle models, software technologies, or development methodologies.

An organization may implement the practices to establish and subsequently improve its capabilities in the acquisition, supply, development, operation, evolution and support of software. The architecture of this model organizes the practices to help software personnel understand and use them for continuous improvement of the management of software processes.

During a software process assessment, the architecture helps the assessor to make judgements about the organization's processes.

Evolving process capability is expressed in terms of capability levels, common features, and generic practices. Generic practices are applicable to all process. They represent the activities necessary to manage a process and improve its capability to achieve desired outputs. They are grouped into common features and, in turn, into capability levels which help define how effective the process will be at achieving its defined purpose. There are six capability levels in the model.

Each process in the model is described by base practices, which are the essential activities of the specific process. Processes are grouped in turn into five process categories.

The base practices may be extended through the generation of application or sector specific practice guides to take account of specific industry, sector, or other requirements.

2 Normative references

There are no normative references in this part og the International Standard

3 Definitions

For the purposes of this part of the International Standard, the definitions in *Software Process Assessment – Part 9: Vocabulary* apply.

4 Nomenclature

4.1 General

A nomenclature for practices is defined in order to identify them unambiguously and relate them to the architecture of the model. The nomenclature for base practices facilitates the identification of process categories, the process that belong to each process category, and the base practices that belong to each process. For generic practices, the nomenclature facilitates the identification of capability levels, the common features that belong to each capability level, and the generic practices that belong to each common feature. In software process assessments using the process model, the nomenclature and identifiers contained in this model shall be used to identify processes to be assessed and to identify the ratings in the assessment output (see part 3 of this International Standard). The nomenclature shall also be used in constructing extended processes and variant models (see Annex A)

4.2 Definition

Each practice is assigned an identifier consisting of a three-character alphanumeric code.

For a base practice, the identifier is of the form PC.PR.PT.

For a generic practice, the identifier is of the form CL.CF.PT.

The codes are:

PC process category identifier

PR process number (within the process category)

CL capability level number

CF common feature number (within the capability level)

PT practice number (within the process or common feature)

For example, “ENG.3.1” denotes a base practice in process category ENG (Engineering), process 3 (Develop software design), base practice 1 “Develop software architectural design”.

Similarly, “2.3.1” denotes a generic practice in capability level 2 (Planned-and-Tracked), common feature 3 (Verifying Performance), generic practice 1 “Verify process compliance”.

5 Generic practices

During an assessment, generic practices, grouped according to common feature and capability level, are used to determine the capability of a process.

The generic practices, defined in the following tables, shall apply to all processes defined within this part of the International Standard.

Table 2 – Generic Practices for Level 1: Performed-Informally

	Common Feature 1.1: Performing Base Practices
1.1.1	**Perform the process.** Perform the base practices of the process to provide work products and/or services to a customer. Note: The customer of the process may be internal or external to the organization.

Table 3. – Generic Practices for Level 2: Planned-and-Tracked

	Common Feature 2.1: Planning Performance
2.1.1	**Allocate resources.** Allocate adequate resources (including people) for performing the process. Note: Resource management is described in project process, "Manage resources and schedule" PRO.7.
2.1.2	**Assign responsibilities.** Assign responsibilities for developing the work products and/or providing the services of the process. Note: Assigning responsibility does not necessarily entail detailed job descriptions. Responsibility could be assigned via living documents, such as a task plan. Dynamic assignment of roles is another legitimate implementation of this practice, so long as there are mechanisms in place to assure that the responsibility is assumed.
2.1.3	**Document the process.** Document the approach to performing the process in standards and/or procedures. Note: Employee participation in developing standards and procedures is essential to creating a usable process definition.
2.1.4	**Provide tools.** Provide appropriate tools to support performance of the process.
2.1.5	**Ensure training.** Ensure that the individuals performing the process are appropriately trained in how to perform the process.
2.1.6	**Plan the process.** Plan the performance of the process. Note: Project planning is described in the process, "Establish project plan" PRO.2. Plans are based on estimates, which implies the use of software measurements. See 2.4.1 “Track with measurement” for the related tracking practice.

Table 3. (concluded) **– Generic Practices for Level 2: Planned-and-Tracked**

	Common Feature 2.2: Disciplined Performance
2.2.1	**Use plans, standards, and procedures.** Use documented plans, standards, and/or procedures in implementing the process. Note: The standards and procedures used were documented in 2.1.3. The plans used were documented in 2.1.6.
2.2.2	**Do configuration management.** Place work products of the process under version control or configuration management, as appropriate. Note: The process for configuration management is described in support process, "Perform configuration management" SUP.2.
	Common Feature 2.3: Verifying Performance
2.3.1	**Verify process compliance.** Verify compliance of the process with applicable standards and/or procedures. Note: The applicable standards and procedures were documented in 2.1.3 and used in 2.2.1. The quality assurance process is described in support process, "Perform quality assurance" SUP.3.
2.3.2	**Audit work products.** Verify compliance of work products with the applicable standards and/or requirements. Note: Requirements are identified in Customer-Supplier process, "Identify customer needs" CUS.3, and managed in project process, "Manage requirements" PRO.4.
	Common Feature 2.4: Tracking Performance
2.4.1	**Track with measurement.** Track the status of the process against the plan using measurement. Note: The use of measurement implies that the measures have been defined and selected, and data has been collected.
2.4.2	**Take corrective action.** Take corrective action as appropriate when progress varies significantly from that planned. Note: Progress may vary because estimates were inaccurate, performance was affected by external factors, or the requirements, on which the plan was based, have changed. Corrective action may involve changing the process or changing the plan or both.

Table 4 – Generic Practices for Level 3: Well-Defined

	Common Feature 3.1: Defining a Standard Process (Organization-Level Common Feature)
3.1.1	**Standardize the process.** Document a standard process or family of processes for the organization, which describes how to implement the base practices for the process. Note: The critical distinction between 2.1.3 and 3.1.1 is the scope of application of the standards and procedures. In 2.1.3, the standards and procedures may be used in only a specific instance of the process, e.g., in a particular project. In 3.1.1, standards and procedures are being established at an organizational level for common use. The process definition process is described in the Organization process category, "Define the process" ORG.2.
3.1.2	**Tailor the standard process.** Tailor the organization's standard process family to create a defined process which addresses the particular needs of a specific use. Note: The phrase “which addresses the particular needs of a specific use” caters to the general case of organization-level, as opposed to project-level, processes. For defined processes at the project level, the tailoring addresses the particular needs of the project. The organization's standard process family is documented in 3.1.1 and ORG.2.
	Common Feature 3.2: Performing the Defined Process
3.2.1	**Use a well-defined process.** Use a well-defined process in implementing the process. Note: The defined process will typically be tailored from the organization’s standard process, as described in 3.1.2. A well-defined process is one with inputs, entry criteria, tasks, validation, outputs, and exit criteria that are documented, consistent, and complete.
3.2.2	**Perform peer reviews.** Perform peer reviews of appropriate work products of the process. Note: The process for peer reviews is described in support process, "Perform peer reviews" SUP.5.
3.2.3	**Use well-defined data.** Use data on performing the defined process to manage the defined process. Note: This is an evolution of 2.4.2; corrective action taken here is based on a well-defined process, which has objective completion (exit) criteria (see 3.2.1) for determining progress.

Table 5 – Generic Practices for Level 4: Quantitatively-Controlled

	Common Feature 4.1: Establishing Measurable Quality Goals
4.1.1	**Establish quality goals.** Establish measurable quality goals for the work products of the organization's standard process family. Note: These quality goals should be tied to the strategic quality goals of the organization, the particular needs and priorities of the customer, and/or to the tactical needs of the project.
	Common Feature 4.2: Objectively Managing Performance
4.2.1	**Determine process capability.** Determine the process capability of the defined process quantitatively. Note: This is a quantitative process capability based on a well-defined (3.1.1) and measured process.
4.2.2	**Use process capability.** Take corrective action as appropriate when the process is not performing within its process capability. Note: Special causes of variation, identified based on an understanding of process capability, are used to understand when and what kind of corrective action is appropriate. This is an evolution of 3.2.3, with the addition of quantitative process capability to the defined process.

Table 6 – Generic Practices for Level 5: Continuously-Improving

	Common Feature 5.1: Improving Organizational Capability (Organization-Level Common Feature)
5.1.1	**Establish process effectiveness goals.** Establish quantitative goals for improving process effectiveness of the standard process family, based on the business goals of the organization and the current process capability.
5.1.2	**Continuously improve the standard process.** Continuously improve the process by changing the organization's standard process family to increase its effectiveness. Note: Changes to the organization's standard process family may come from innovations in technology or incremental improvements. Innovative improvements will usually be externally driven by new technologies. Incremental improvements will usually be internally driven by improvements identified during tailoring (3.1.2) or defect prevention (5.2.2) activities. The approach to improving the standard process is to attack common causes of variation. Special causes of variation are controlled in 4.2.2. Common causes of variation across instantiations of the process are changed in this practice.
	Common Feature 5.2: Improving Process Effectiveness
5.2.1	**Perform causal analysis.** Perform causal analysis of defects.
5.2.2	**Eliminate defect causes.** Eliminate the causes of defects in the defined process selectively. Note: Defect causes are selectively eliminated because it may be impractical to perform causal analysis (5.2.1) on all defects, so some screening criteria may be used. These criteria for prioritizing defect removal should be identified and documented. Also note the desirability of eliminating similar, as yet undiscovered, defects in the product, as well as eliminating the cause of the defect.
5.2.3	**Continuously improve the defined process.** Continuously improve process performance by changing the defined process to increase its effectiveness. Note: The improvements may be based on incremental improvements (5.2.2) or new technologies (perhaps as part of pilot testing in ORG.3.8.) Improvements will typically be driven by the goals established in 5.1.1.

6 Base practices

In an assessment conducted according to the provisions of this International Standard, the processes included within the scope of the assessment shall be mapped to one or more of the processes defined in this clause, or to a variant model constructed and documented in accordance with the requirements in Annex A..

The assessment shall include all of the base practices of each process within the scope of the assessment.

The rest of this clause lists all the base practices organized as a hierarchy, first by process category, then by process. All process categories, processes, and base practices are given both a name and description.

The five process categories are:

CUS Customer - Supplier

ENG Engineering

PRO Project

SUP Support

ORG Organization

The description of each process category includes a characterization of the processes it contains, followed by a list of process names.

The first paragraph in the description of each process states the purpose of the process. Subsequent paragraphs, if present, clarify what is said in the purpose, describe inputs/outputs to other processes, and describe when the process is invoked.

Base practices follow the process description. Each base practice is numbered (see clause 4) to allow easy identification and reference. This number and name of the base practice appear in bold. Immediately following is a statement of what the practice does. This is sometimes followed by a note which provides an example, a more detailed explanation, or a cross-reference note.

6.1 Customer-Supplier process category (CUS)

The *Customer-Supplier* process category consists of processes that directly impact the customer, supporting development and transition of the software to the customer, and provide for its correct operation and use.

Note: Throughout the base practices, "customer" can mean either an external customer or an internal customer.

The processes belonging to the Customer-Supplier process category are:

CUS.1 Acquire software product and/or service

CUS.2 Establish contract

CUS.3 Identify customer needs

CUS.4 Perform joint audits and reviews

CUS.5 Package, deliver, and install the software

CUS.6 Support operation of software

CUS.7 Provide customer service

CUS.8 Assess customer satisfaction

CUS.1 Acquire software product and/or service

The purpose of the *Acquire software product and/or service* process is to define the activities that must be performed by the customer or the acquirer to obtain the software product or service. The acquirer is the party that obtains the software product and/or service from the supplier. In some cases the acquirer and the customer (the party who will utilize the software product and/or service) may be the same party. In other cases, where there exists a separate party (organization) that is solely performing the acquisition duties, the acquirer and the customer will not be the same party.

The activities involved in this process include the definition of the need to acquire a software product and/or service through to the proposal, supplier selection, and product and/or service acceptance.

Note: This process would typically be applied in an assessment when the software organization being assessed is acting as a customer to a software subcontractor or vendor who is supplying it with software products or services.

CUS.1.1 Identify the need. Identify a need to acquire, develop, or enhance a software product.

Note: The need may be necessitated by a number of circumstances including: business, regulatory, research, safety, security.

CUS.1.2 Define the requirements. Prepare the system and software requirements to satisfy the need for a new product and/or service.

Note: This definition of the requirements may be done completely or partially by the supplier. See "Develop system requirements and design" ENG.1, and "Develop software requirements" ENG.2.

Also see "Obtain customer requirements and requests" CUS.3.1. CUS.1.2 is focusing on defining requirements when the software organization is acting as a customer. CUS.3.1 is focusing on obtaining requirements when the software organization is acting as a supplier. The primary difference is one of perspective, depending on the role being performed.

CUS.1.3 Prepare acquisition strategy. Prepare a strategy for the acquisition of the product including:

- make/buy risk analysis (off-the-shelf, develop internally, develop through contract, enhance existing software product);
- acceptance strategy.

CUS.1.4 Prepare request for proposal. Prepare a request for proposal tender including acquisition requirements and project schedule.

CUS.1.5 Select software product supplier. Select a supplier for the acquired software product and/or service based upon an evaluation of supplier proposals, capabilities and other factors which may be particular to the product.

Note: After the supplier has been chosen, a contract is established between the customer and the supplier. See process, "Establish contract" CUS.2. Monitoring of the supplier and acceptance of the product are performed through the process, "Perform joint audits and reviews" CUS.4. Management of subcontracted work is performed by "Manage subcontractors" PRO.8.

CUS.2 Establish contract

The purpose of the *Establish contract* process is to develop a contract which clearly expresses the expectations, responsibilities, and liabilities of both the supplier and the customer.

Note: Process, "Perform joint audits and reviews" CUS.4 addresses joint audits and reviews of this contract.

CUS.2.1 Review before contract finalization. Review the contents of the contract prior to its finalization.

Note: This review typically includes:

- scope of contract and requirements;
- possible contingencies or risks;
- alignment of the contract with the strategic business plan of the organization;
- protection of proprietary information;
- requirements which differ from those in the original documentation;
- capability to meet contractual requirements;
- responsibility for subcontracted work;
- terminology;
- the customer ability to meet contractual obligations.

CUS.2.2 Negotiate contract. Negotiate a contract with the customer.

Note: This contract typically includes

- schedules for product delivery;
- terms of payment;
- customer's acceptance criteria;
- procedures for handling changes in customer requirements;
- procedures for handling customer requests for process/product quality monitoring;
- procedures for handling customer-detected problems;
- customer's role in the development and maintenance process;
- resources to be provided by the customer;
- standards and procedures to be used;
- servicing and maintenance requirements.

CUS.2.3 Determine interfaces to independent agents. Determine the supplier and customer interfaces to independent verification, validation, and/or test agents, and document in the contract.

CUS.2.4 Determine interfaces to subcontractors. Determine the supplier and customer interfaces to other parties, such as subcontractors, who will be involved in the work described in the contract, or whose work will impact its success; document in the contract.

CUS.3 Identify customer needs

The purpose of the *Identify customer needs* process is to manage the gathering, processing, and tracking of customer requirements and requests toward a better understanding of what will satisfy the customer.

Compare with process, "Manage requirements" PRO.4, which addresses coming to an understanding within the software project of the requirements to build to (and which become part of the development baseline). Instead, the current process emphasizes interaction with the customer to promote an understanding of customer requirements, and tracks all requirements and requests.

CUS.3.1 Obtain customer requirements and requests. Obtain customer requirements and requests through direct solicitation of customer and user input and through review of: customer business proposals, target hardware environment, and other documents bearing on customer requirements.

CUS.3.2 Understand customer expectations. Review with customers and users their requirements and requests to better understand their needs and expectations.

Note: This may include joint meetings with the customer and as appropriate, models and prototypes. The performance of this base practice may coincide with base practice, "Evaluate requirements with customer" ENG.2.4.

CUS.3.3 Keep customers informed. Keep customers informed about the status and disposition of their requirements and requests.

Note: This may include joint meetings with the customer or formal communication to review the status for their requests including requirements.

CUS.4 Perform joint audits and reviews

The purpose of the *Perform joint audits and reviews* process is to maintain a common understanding with the customer of the progress against the objectives of the contract and what should be done to help ensure development of a product that satisfies the customer.

The work of this process is accomplished through different kinds of audits and reviews, including: contract audits, management reviews, technical reviews, and the acceptance review. Many of these reviews will be scheduled to coincide with development milestones and other milestones specified in the contract. The process "Establish contract" CUS.2, deals with contract reviews.

CUS.4.1 Establish joint reviews and audits. Establish which joint reviews and audits will be performed with the customer.

CUS.4.2 Prepare for customer audits and reviews. Prepare for customer audits and reviews by establishing

- scope of review;
- a checklist for the review;
- the desired outputs;
- a schedule;
- who should attend;
- an approach to problem identification and resolution;
- facility needs.

CUS.4.3 Conduct joint management reviews. Conduct regular joint management reviews with the customer to discuss and assess

- proposal against the requirements;
- status against project plans;
- schedules;
- risks;
- compliance with appropriate standards;
- readiness for the next development steps.

CUS.4.4 Conduct joint technical reviews. Conduct regular joint technical reviews with the customer to discuss technical issues and assess technical status against customer requirements and acceptance criteria documented in the contract.

CUS.4.5 Support customer acceptance review. Support the customer in his evaluation of the software product, providing evidence that the software product is complete and correct, complies with appropriate standards and specifications, and satisfies the acceptance criteria documented in the contract.

Note: This practice may include acceptance testing by the customer.

CUS.4.6 Perform joint process assessment. Conduct regular joint software process assessments with the customer to jointly review both the organization's processes along with interfaces with the customer processes.

CUS.5 Package, deliver, and install the software

The purpose of the *Package, deliver, and install the software* process is to package, deliver, and install the software at the customer site to ensure its effective operation, handling, and storage. The base practices in this process category are critical to preserving the quality of the software and all associated deliverables with the software.

CUS.5.1 Identify installation requirements. Identify requirements for packing, delivering, and installing the software, addressing as appropriate

- type of media;
- documentation;
- copyrights and licensing;
- custody of master and backup copies;
- provision for copying;
- critical safety and security issues.

CUS.5.2 Prepare site for installation. Prepare the site for the installation of the software product.

Note: This may be performed by the customer.

CUS.5.3 Pack software. Label and pack the software and its accompanying documentation, including a packing list identifying the contents along with information such as what is new or changed in the software.

Note: The software product will be created through base practice, "Build product releases" SUP.2.6 and will typically identify release/version numbers.

CUS.5.4 Deliver software. Deliver the software, excluding any non-deliverable items used during its development or maintenance.

CUS.5.5 Verify correct receipt. Verify the correctness and completeness of the delivered software, including the released package, delivery instructions and associated documentation.

Note: This base practice will be usually carried out with or by the customer.

CUS.5.6 Install software. Install the software at the customer site, recording the steps taken and the results.

CUS.5.7 Provide handling and storage procedures. Define and provide procedures for handling and storing the software and its documentation including

- providing for master copies of code and documentation;
- disaster recovery;
- addressing appropriate critical safety and security issues.

CUS.6 Support operation of software

The purpose of the *Support operation of software* process is to support the correct and effective operation of the software for the duration of its intended operation.

This process assumes that the software has already been installed (see process, "Package, deliver, and install the software" CUS.5) and is operating as part of a larger system, with an operator responsible for ensuring continuing operation of the system, and users who use the system.

The management of changes to the software and system to support the software operation will be accomplished by process, "Maintain system and software" ENG.7.

One possible input to this process would be as the quality requirements for the software operation determined by base practice, "Determine software requirements" ENG.2.1.

Note: Some practices may be performed by a different party. For example, the operator may be the customer, the developer, or a third party (see, "Operate the software" CUS.6.3). The operator is the individual/organization whose responsibility it is to ensure operation of the software system. The user is the individual/organization who utilizes the software system to perform its tasks.

CUS.6.1 Identify operational risks. Identify and mitigate risks to system operation and functionality that are due to such factors as: environmental failure, hardware or software failure, or network failure.

CUS.6.2 Perform operational testing. Perform operational testing of each release of the software, assessing satisfaction against specified criteria.

CUS.6.3 Operate the software. Operate the software in its intended environment and in the specified way.

CUS.6.4 Resolve operational problems. Identify, record, and resolve problems arising from operation of the software (i.e. problems encountered by the operator as opposed to a user).

Note: The elimination of the cause of the operational problem will be handled by base practice, "Correct the defect" SUP.4.6.

CUS.6.5 Handle user requests. Monitor, record, and respond to all user requests and problems relating to the software, forwarding as appropriate to the maintenance function.

Note: This ties to process, "Maintain system and software" ENG.7.

CUS.6.6 Document temporary work-arounds. Provide documented temporary work-arounds as appropriate to maintain operation of the system until a permanent solution to a problem can be found.

Note: Keep the customer informed of the status and availability of the permanent solution.

CUS.6.7 Monitor system capacity and service. Provide the capability to monitor system capacity and operational service on a regular basis.

CUS.7 Provide customer service

The purpose of the *Provide customer service* process is to establish and maintain an acceptable level of service to the customer to support effective use of the software.

CUS.7.1 Train customer. Provide training and documentation, as appropriate, to the customer so that the software can be effectively used.

CUS.7.2 Establish product support. Establish a service by which the customer can raise problems and questions encountered in use of the software, and receive help in resolving them.

CUS.7.3 Monitor performance. Monitor the software performance in order to be aware of performance problems which might impact level of service.

CUS.7.4 Install product upgrades. Plan, test, and install modifications to the software and the documentation to correct defects or improve performance of the software for the customer, and thus maintain or improve the level of service.

Note: This ties to process, "Maintain system and software" ENG.7 and process, "Package, deliver, and install the software" CUS.5.

CUS.8 Assess customer satisfaction

The purpose of the *Assess customer satisfaction* process is to determine the level of customer satisfaction with the software and services received (operation, customer support).

CUS.8.1 Determine customer satisfaction level. Determine the level of customer satisfaction with the software products and services received through, as appropriate, field performance data, surveys, interviews, and studies.

Note: In some instances the end-user of the software may be different from the customer of the software. In this case, both the customer and end-user satisfaction levels should be determined.

CUS.8.2 Compare with competitors. Compare the level of customer satisfaction obtained for your software and services received relative to that of your competitors.

Note: It may be necessary to obtain information on your competitors from third party sources. It may also be necessary to include information on how competitors define customer satisfaction, measurement techniques, criteria, collection and evaluation methods, etc., to provide a meaningful comparison.

CUS.8.3 Communicate customer satisfaction. Communicate customer satisfaction data throughout the organization.

6.2 Engineering process category (ENG)

The *Engineering* process category consists of processes that directly specify, implement, or maintain a system and software product and its user documentation.

In some circumstances, there is no "system" so the scope of the engineering processes is reduced to only software and user documentation, and processes ENG.1 and ENG.6 become "not applicable."

While the processes listed below appear in "waterfall model" sequence, the intent is not to preclude either their concurrent or iterative execution. (The sequence is determined and documented by base practice, "Determine release strategy" ENG.1.4 and by process, "Plan project life cycle" PRO.1.)

Inputs to the "Engineering" process category possibly include a contract or agreement describing what work is to be done, and a plan(s) on how that is to be accomplished (see processes, "Establish contract" CUS.2, and "Establish project plan" PRO.2.)

The processes belonging to the "Engineering" process category are:

ENG.1 Develop system requirements and design

ENG.2 Develop software requirements

ENG.3 Develop software design

ENG.4 Implement software design

ENG.5 Integrate and test software

ENG.6 Integrate and test system

ENG.7 Maintain system and software

ENG.1 Develop system requirements and design

The purpose of the *Develop system requirements and design* process is to establish the system requirements and system design, identifying which system-level requirements should be allocated to which elements of the system design and to which releases of the system.

Note: This process will typically not be performed by a software group, but the group which does perform it should include a member(s) with software expertise.

ENG.1.1 Specify system requirements. Determine the required functions and capabilities of the system and document in a system requirements specification.

Note: the system requirements specification describes such things as

- functions and capabilities of the system;
- performance of the system;
- safety;
- reliability;
- security;
- human engineering;
- interface;
- operations, and maintenance requirements;
- design constraints and qualification requirements.

See CUS.3 for discussion of customer requirements used as an input to system requirements analysis.

ENG.1.2 Describe system architecture. Establish the top-level system architecture, identifying elements of

- hardware;
- software;
- manual operations.

ENG.1.3 Allocate requirements. Allocate all system requirements to the elements of the top-level system architecture, including software.

Note: The result of performing base practices ENG.1.2 and ENG.1.3 is a documented product configuration which describes the position of each element in the system architecture and the requirements which it must address.

ENG.1.4 Determine release strategy. Prioritize the system requirements and map them to future releases of the system.

Note: Rather than wait to release a product in which all requirements are achieved, it may instead be desirable to prioritize the system requirements and to release a sequence of products which address increasing subsets of the prioritized requirements, e.g. to establish market share early. A possible input to this base practice is the project's software life cycle model, produced by process, "Plan project life cycle" PRO.1, or the equivalent at the system level.

ENG.2 Develop software requirements

The purpose of the *Develop software requirements* process is to establish, analyze and refine the software requirements.

ENG.2.1 Determine software requirements. Determine the software requirements and document in a software requirements specification.

Note: The software requirements specification describes such things as

- functions to be performed and their performance characteristics;
- software interfaces (to hardware, operating system, and user) and their characteristics;
- reliability characteristics;
- installation and maintenance requirements;
- safety requirements;
- security characteristics.

In addition, there is value in specifying requirements, particularly quality requirements, in quantitative terms, so that an objective evaluation of their satisfaction can later be made.

ENG.2.2 Analyze software requirements. Analyze the software requirements for correctness.

Note: Aspects of correctness to analyze include

- completeness;
- understandability;
- testability;
- verifiability;
- feasibility;
- validity;
- consistency;
- adequacy of content.

Depending on the software life cycle model chosen, it may be desirable to have only a select set of requirements "correct" (implemented), leaving the others to be addressed in subsequent iterations of this process. See base practice, "Update requirements for next iteration" ENG.2.5, below.

ENG.2.3 Determine operating environment impact. Determine the impact of the software requirements on the operating environment.

Note: The operating environment includes tasks performed by or other systems used by the intended uses of the software product.

ENG.2.4 Evaluate requirements with customer. Communicate the software requirements to the customer, and revise if necessary, based on what is learned through this communication.

Note: Prototyping or simulation may be appropriate methods of evaluating the requirements with the customer. The performance of this base practice may coincide with base practice, "Understand customer expectations" CUS.3.2.

ENG.2.5 Update requirements for next iteration. After completing an iteration of requirements, design, code, and test, use the feedback obtained from use to modify the requirements for the next iteration.

ENG.3 Develop software design

The purpose of the *Develop software design* process is to establish a software design that effectively accommodates the software requirements; at the top-level this identifies the major software components and refines these into lower level software units which can be coded, compiled, and tested.

ENG.3.1 Develop software architectural design. Transform the software requirements into a software architecture that describes the top-level structure and identifies its major components.

ENG.3.2 Design interfaces at top level. Develop and document a top-level design for the external and internal interfaces.

ENG.3.3 Develop detailed design. Transform the top-level design into a detailed design for each software component. The software components are refined into lower levels containing software units that can be coded, compiled, and tested.

Note: The detailed design includes the specification of external and internal interfaces between the software units.

The result of this base practice is a documented software design document which describes the position of each software unit in the software architecture and the functional, performance, and quality characteristics which each must address.

ENG.3.4 Establish traceability. Establish traceability between the software requirements and the software designs.

ENG.4 Implement software design

The purpose of the *Implement software design* process is to produce executable and independently tested units of software code which implement the components of the software design.

ENG.4.1 Develop software units. Develop and document each software unit, including

- the code;
- data structures;
- database.

Note: This base practice involves creating, documenting, and compiling representations of each software unit using expressions in the appropriate programming language(s).

ENG.4.2 Develop unit verification procedures. Develop and document procedures for verifying that each software unit satisfies its design requirements.

Note: The normal verification procedure will be through unit testing, and the verification procedure will include unit test cases and unit test data.

ENG.4.3 Verify the software units. Verify that each software unit satisfies its design requirements and document the results.

ENG.5 Integrate and test software

The purpose of the *Integrate and test software* process is to integrate the software units with each other producing software that will satisfy the software requirements.

This process is accomplished through developing aggregates of software units and testing them as an aggregate, and then testing the resulting integrated software to ensure it satisfies the software requirements.

Note: 1) Testing is normally done by individuals or teams independent of the developers. 2) Software test planning should be started early, e.g. at the same time as developing the software requirements and design, to allow for adequate test preparation.

ENG.5.1 Determine regression test strategy. Determine the conditions for retesting aggregates against their tests should a change in a given software unit be made.

ENG.5.2 Build aggregates of software units. Identify aggregates of software units and a sequence or partial ordering for testing them.

Note: Typically, the software architecture and the release strategy will have some influence on the selection of aggregates.

ENG.5.3 Develop tests for aggregates. Describe the tests to be run against each software aggregate, indicating input data and acceptance criteria.

ENG.5.4 Test software aggregates. Test each software aggregate ensuring that it satisfies the test criteria, and document the results.

ENG.5.5 Develop tests for software. Describe the tests to be run against the integrated software, indicating software requirements being checked, input data, and acceptance criteria.

Note: Tests can be developed during process, "Develop software requirements" ENG.2, “Develop software design” ENG.3, and “Implement software design” ENG.4. Test development should not wait until software integration, covered in base practices ENG.5.2-5.4).

The set of tests should demonstrate compliance with the software requirements and provide coverage of the internal structure of the software.

ENG.5.6 Test integrated software. Test the integrated software ensuring that it satisfies the software requirements, and document the results.

ENG.6 Integrate and test system

The purpose of the *Integrate and test system* process is to integrate the software with the manual operations and hardware elements producing a system that will satisfy the system requirements.

This process is accomplished through developing aggregates of system elements and testing them as an aggregate, and then testing the resulting integrated system to ensure it satisfies the system requirements.

Note: 1) This process will typically not be performed by a software group, but the group which does perform it should include a member(s) with software expertise.

2) System test planning should be started early, e.g. about the same time as developing the system requirements and design, to allow for adequate test preparation.

ENG.6.1 Build aggregates of system elements. Identify aggregates of system elements and a sequence or partial ordering for testing them.

Note: Typically, the system architecture and the release strategy will have some influence on the selection of aggregates.

ENG.6.2 Develop tests for aggregates. Describe the tests to be run against each system aggregate, indicating input data, system components needed to perform the test, and acceptance criteria.

ENG.6.3 Test system aggregates. Test each system aggregate ensuring that it satisfies its requirements, and document the results.

ENG.6.4 Develop tests for system. Describe the tests to be run against the integrated system, indicating system requirements being checked, input data, and acceptance criteria.

Note: 1) This can be performed during process, "Develop system requirements and design" ENG.1 (it should not wait until integration, covered in base practices ENG 6.1-6.3 above).

2) The set of tests should demonstrate compliance with the system requirements.

3) For some products, it may be appropriate to conduct extensive field testing.

ENG.6.5 Test integrated system. Test the integrated system ensuring that it satisfies the system requirements, and document the results.

ENG.7 Maintain system and software

The purpose of the *Maintain system and software* process is to modify the system, its hardware, the network system (if any), software, and associated documentation in response to user requests while preserving the integrity of the system design.

There are several sources which create the need for modifying the system or software. Example sources include

- detected error;
- deficiency;
- problem in the operation of the system or software;
- particular improvement or modification of the system or software required or requested by the customer (internal or external).

Note: This process interacts closely with several customer processes and their base practices, and may be even partially subsumed by them, for example

- "Support operation of software" CUS.6;
- its base practice, "Handle user requests" CUS.6.6;
- "Provide customer service" CUS.6;
- its base practice, "Install product upgrades" CUS.7.6.

ENG.7.1 Determine maintenance requirements. Determine the system and software maintenance requirements, identifying the system and software elements to be maintained, and their required enhancements.

Note: Some of the required enhancements may have been previously planned but deferred.

ENG.7.2 Analyze user problems and enhancements. Analyze user problems and requests and required enhancements, evaluating the possible impact of different options for modifying the operational system and software, system interfaces, and requirements.

Note: Several base practices address the collection and tracking of user problems and requests

- "Handle user requests" CUS.6.6;
- "Obtain customer requirements and requests" CUS.3.1;
- base practices in process, "Perform problem resolution" SUP.4.

ENG.7.3 Determine modifications for next upgrade. Based on the above analyses, determine which modifications should be applied in the next system or software upgrade, documenting which software units and other system elements and which documentation will need to be changed and which tests will need to be run.

ENG.7.4 Implement and test modifications. Use the other engineering processes, as appropriate, to implement and test the selected modifications, demonstrating that the unmodified system and software requirements will not be compromised by the upgrade.

ENG.7.5 Upgrade user system. Migrate the upgraded system and software with applied modifications to the user's environment, providing for, as appropriate

- parallel operation of the previous and upgraded systems;
- additional user training;
- support options;
- retirement of the previous system.

6.3 Project process category (PRO)

The *Project* process category consists of processes which establish the project, and co-ordinate and manage its resources to produce a product or provide services which satisfy the customer.

The input is a contract or agreement to do the work (see process, "Establish contract" CUS.2). The focus in this process category is on the effective use of resources (time, effort, people, money) toward accomplishing the purpose and objectives of the project.

The processes belonging to the "Project" process category are:

PRO.1 Plan project life cycle

PRO.2 Establish project plan

PRO.3 Build project teams

PRO.4 Manage requirements

PRO.5 Manage quality

PRO.6 Manage risks

PRO.7 Manage resources and schedule

PRO.8 Manage subcontractors

PRO.1 Plan project life cycle

The purpose of the *Plan project life cycle* process is to establish an appropriate software life cycle model for the project.

The input to this process is an agreement or contract to develop a software product, which identifies the project objectives with regard to product quality, cost, and/or schedule.

The output of this process is a software life cycle model with descriptions of software activities and tasks to be performed by the project and identification of project controls (management and technical reviews, etc.).

PRO.1.1 Evaluate options for product development. Evaluate options for product development, identifying the risks associated with each.

Note: Example product development options include

- using internal resources;
- subcontracting;
- using off-the-shelf products;
- using customer-furnished products;
- combination of all four.

PRO.1.2 Select software life cycle model. Select a software life cycle model for the project which is appropriate to the scope, magnitude, and complexity of the project.

Note: Examples of software life cycle models include waterfall, spiral and serial build.

Note that the life cycle might be selected by the customer or by contract terms.

PRO.1.3 Describe activities and tasks. Describe the project's software activities and their associated tasks, and purpose of each.

PRO.1.4 Establish task sequence. Establish the software task sequence or partial ordering within the identified software life cycle, identifying where the following occur

- management and technical reviews;
- software audits;
- peer reviews.

PRO.1.5 Document activities. Identify and document software development activities and tasks, including inputs, outputs, and interfaces.

PRO.2 Establish project plan

The purpose of the *Establish project plan* process is to establish reasonable plans for performing the software engineering and to form a basis for managing the software project.

Project plans typically document

- project purpose and objectives;
- work products to be developed;
- software life cycle model;
- software estimates;
- project risks and mitigation plans;
- schedule;
- resources allocated to the project activities.

Inputs to this process can include the project's software life cycle model produced by process, "Plan project life cycle" PRO.1.

PRO.2.1 Develop work breakdown structure. Develop a work break down structure relating project tasks and sequence with the resources required to accomplish them.

PRO.2.2 Identify project standards. Identify the software standards which will guide the project's software activities, consistent with the needs of the project.

PRO.2.3 Identify specialized facilities. Identify any specialized tools, equipment, or rooms beyond those normally available which will be needed to meet any unusual technical requirements of the project.

Note: Examples include

- target hardware;
- simulators;
- specialized test equipment;
- test laboratories.

The process, "Provide software engineering environment" ORG.6 deals with providing the (normal) software engineering environment. Specialized facilities may be needed due to security requirements of the project.

PRO.2.4 Determine reuse strategy. Identify opportunities for reuse, analyze their impacts, and determine the strategy for reuse.

Note: This base practice benefits from process, "Enable reuse" ORG.5.

PRO.2.5 Develop project estimates. Develop estimates of what is needed to satisfy the software requirements for the entire software life cycle.

Note: Parameters to estimate include

- size;
- effort;
- cost;
- schedule;
- resources.

PRO.2.6 Identify initial project risks. Identify, assess, and document an initial (or baseline) set of project risks and their mitigation plans.

Note: Software project risks include

- risks to staying within budgeted software size, cost, effort, schedule;
- risks in availability of resources;
- technical risks.

See the process, "Manage risks" PRO.6.

PRO.2.7 Identify project measures. Identify the basic set of status and other measures which will be used to track project progress and help determine if the project is meeting its objectives.

PRO.2.8 Establish project schedule. Establish the project schedule, based on the software life cycle model, work breakdown structure, estimates, and risk mitigation plans.

PRO.2.9 Establish project commitments. Establish commitments to the estimates and plans with all affected groups.

PRO.2.10 Document project plans. Document the results of the activities in this process within the project plans.

Note: This includes documenting the project's software life cycle model.

PRO.3 Build project teams

The purpose of the *Build project teams* process is to establish project teams with qualified members who can fulfil their responsibilities on their team and work together as a cohesive group.

The term "team" is intended to cover a number of different kinds and durations of teams, including

- mixed teams consisting of both customer and supplier representatives;
- review teams, development teams, problem analysis teams, process improvement teams etc.;
- teams which exist for a short duration to solve a specific problem, or long-term as an established part of the environment.

A likely input is the software project plans from process, "Establish project plan" PRO.2.

PRO.3.1 Define project teams. Define the teams which will be needed to perform the work of the project, defining the structure and operating rules for the team, required knowledge and skills.

PRO.3.2 Empower project teams. Empower teams to perform their job, by ensuring that they have

- an understanding of their job;
- a shared vision or sense of common interest;
- appropriate mechanisms or facilities for communication and work;
- support from the appropriate management for what they are trying to accomplish;

PRO.3.3 Maintain project team interactions. Obtain and maintain agreement on the implementation of interactions between teams.

Note: Example areas on which to agree include

- responsibilities;
- commitments;
- interfaces;
- interaction methods;
- conflict resolution methods.

PRO.3.4 Manage inter-team issues. Identify, track, and resolve issues that affect the progress of more than one team or threaten project unity.

PRO.4 Manage requirements

The purpose of the *Manage requirements* process is to establish a software requirements baseline, which serves as the basis for the project's software work, products, and activities; and manage changes to that baseline.

Note: A key difference between this process and "Identify customer needs" CUS.3, is that in CUS.3, the managed requirements are not necessarily specific to a particular project or to just software; here they are specific to both a particular project and to software. See also processes, "Establish contract" CUS.2, and "Develop software requirements" ENG.2, for other processes addressing the requirements.

PRO.4.1 Agree on requirements. Obtain agreement across teams on the customer's requirements, obtaining the appropriate sign-offs by representatives of all teams and other parties contractually bound to work to these requirements.

PRO.4.2 Establish customer requirements baseline. Document the customer's requirements and establish as a baseline for project use.

PRO.4.3 Manage customer requirements changes. Manage all changes made to the customer requirements to ensure those who are affected by the changes are able to assess the impact and risks, and initiate appropriate change control and mitigation actions.

PRO.4.4 Use customer requirements. Use the customer requirements as the basis for

- software project plans;
- requirements specifications;
- work products and activities.

PRO.4.5 Maintain traceability. Establish and maintain traceability of requirements to the project's work products throughout the software life cycle.

PRO.5 Manage quality

The purpose of the *Manage quality* process is to manage the quality of the project's products and services to ensure the resulting products and services satisfy the customer.

Managing quality involves identifying the required quality characteristics of the project's products, working to achieve this quality, and demonstrating that this quality was achieved.

Inputs are the customer requirements and selected elements of the software project plans (see process PRO.2). Outputs should be integrated into the software project plans.

Note: There is another process with a similar name, "Perform quality assurance" SUP.3. Process PRO.5 focuses on identifying what needs to be done to build quality into the products and establishing management controls to ensure this gets done; whereas SUP.3 focuses more on an audit and review approach and on ensuring compliance.

PRO.5.1 Establish quality goals. Based on the customer's requirements for quality, establish quality goals for various checkpoints within the project's software life cycle (e.g. at the end of each phase).

PRO.5.2 Define quality metrics. Define metrics that measure the results of project activities to help assess whether the relevant quality goals have been achieved.

PRO.5.3 Identify quality activities. For each quality goal, identify activities which will help achieve that quality goal and integrate these activities into the software life cycle model.

PRO.5.4 Perform quality activities. Perform the identified quality activities.

PRO.5.5 Assess quality. At the identified checkpoints within the project's software life cycle, apply the defined quality metrics to assess whether the relevant quality goals have been achieved.

PRO.5.6 Take corrective action. When quality goals are not achieved, take corrective action.

Note: The corrective action can involve fixing the product generated by a particular project activity or changing the planned set of activities in order to better achieve the quality goals or both.

PRO.6 Manage risks

The purpose of the *Manage risks* process is to continuously identify and mitigate the risks in a project throughout the life-cycle of a project.

Managing risks involves continuously identifying new risks, working to effectively mitigate these risks, and evaluating the success of risk mitigation efforts.

A few base practices in other processes are likely to be more effective if this process is broadly performed. For example, base practices

- "Plan against failure" CUS.6.2;
- "Identify initial project risks" PRO.2.5.

PRO.6.1 Establish risk management scope. Determine the scope of risk management to be performed for this project: including the severity, probability, and type of risks to identify and manage.

PRO.6.2 Identify risks. Identify risks to the project as they develop.

Note: Risks include cost, schedule, effort, resource, and technical risks.

PRO.6.3 Analyze and prioritize risks. Assess the probability of occurrence, impact, time-frame, causes and interrelationships of risks for determining the priority in which to apply resources to mitigate these risks.

PRO.6.4 Develop mitigation strategies. Define appropriate strategies to take to mitigate each risk or set of related risks.

PRO.6.5 Define risk metrics. For each risk (or set of related risks) define the metrics that measure the change in the risk state (probability, impact, time-frame) and the progress of mitigation activities.

PRO.6.6 Implement mitigation strategies. Carry out the defined mitigation strategies.

PRO.6.7 Assess results of mitigation strategies. At identified checkpoints, apply the defined metrics to assess the expected progress and level of success of the mitigation strategies.

PRO.6.8 Take corrective action. When expected progress is not achieved, take corrective action.

Note: Corrective action may involve developing and implementing new mitigation strategies or adjusting the existing strategies.

PRO.7 Manage resources and schedule

The purpose of the *Manage resources and schedule* process is to co-ordinate and manage the project's resources and schedule during the life of the project toward the end of achieving the project's objectives and software requirements.

The project plans developed in process PRO.2 are inputs to and revised by this process.

PRO.7.1 Acquire resources. Allocate and distribute appropriate and sufficient resources to the software project activities, including both technical and management resources.

PRO.7.2 Track progress. Regularly compare and report the status of the project against the project plans.

Note: Particular aspects of the project to address include

- size;
- effort;
- cost;
- schedule;
- resources.
- risks

PRO.7.3 Conduct management reviews. Regularly and at major milestones, conduct management reviews to discuss and assess

- status against project plans and schedules;
- status of software risks;
- compliance with appropriate standards;
- and readiness for the next development steps.

PRO.7.4 Conduct technical reviews. Regularly and at major milestones, conduct technical reviews to discuss and assess

- technical status against the plans;
- technical issues not yet resolved;
- and technical readiness for the next development steps.

PRO.7.5 Manage commitments. Manage commitments to the estimates and plans with all affected groups, taking action when appropriate.

PRO.8 Manage subcontractors

The purpose of the *Manage subcontractors* process is to select qualified subcontractor(s) and manage their performance.

The base practices of this process help ensure that the supplier and supplier's subcontractors have the same understanding of project objectives and customer requirements and how they will work to jointly meet these successfully.

PRO.8.1 Establish statement of work. Establish a statement of the work to be performed under subcontract.

PRO.8.2 Qualify potential subcontractors. Qualify potential subcontractors through an assessment of their capability to perform the required software function.

Note: Subcontractors may be performing a variety of software related tasks including: software development, maintenance, documentation and training.

PRO.8.3 Select subcontractor. Select qualified subcontractors to perform defined portions of the contract.

PRO.8.4 Establish and manage commitments. Establish and manage commitments from and to the subcontractor.

PRO.8.5 Maintain communications. Exchange information on technical progress regularly with the subcontractor in order to maintain a common understanding of the work being performed.

PRO.8.6 Assess compliance. Assess compliance of the subcontractor against the agreed upon standards and procedures.

PRO.8.7 Assess subcontractor quality. Assess the quality of the subcontractors' delivered products and services to ensure the completeness, correctness and compliance with standards and specifications.

6.4 Support process category (SUP)

The *Support* process category consists of processes which may be employed by any of the other processes (including other supporting processes). The supporting processes can be employed at various life-cycle points and may be performed by the organization employing them, the customer, or by an independent organization.

To employ a supporting process within another process may require some tailoring of the support process, e.g. the formality and rigor of configuration management depends on the work product and severity of need for control and stability of the work product.

When assessing a particular support process, special consideration should be given to whether it has been implemented as broadly as it should be implemented. Also, it will require the assessor's judgement as to whether the implementation is satisfactorily formal and rigorous for the particular needs of the situation.

The processes belonging to the "Support" process category are:

SUP.1 Develop documentation

SUP.2 Perform configuration management

SUP.3 Perform quality assurance

SUP.4 Perform problem resolution

SUP.5 Perform peer reviews

SUP.1 Develop documentation

The purpose of the *Develop documentation* process is to develop and maintain documents needed by managers, engineers, users, or customers of the system or software.

Note: This process covers the development of documents such as

- project management documentation, such as plans;
- engineering work product documentation, such as design rationale;
- process documentation, such as a peer review procedure;
- and end user documentation, which describes the intended use of the system and software to a user.

SUP.1.1 Determine documentation requirements. Identify the requirements for the document to be built, including

- title;
- audience;
- purpose;
- objectives to be achieved;
- outline of its content;
- media and distribution requirements.

Note: The schedule for project documentation should be identified and integrated into the project's software plans.

SUP.1.2 Develop document. Develop the identified document according to its requirements.

SUP.1.3 Check document. Check the completed document against its requirements using, as appropriate, audience representatives, subject matter experts, or documentation experts. Revise the documentation.

Note: The value of performing this base practice is variable and depends on the audience and purpose. In the case of user documentation, this is a particularly important base practice, because it is important that documentation intended for use by system and software users adequately describe the system and software and how it is to be used in a manner which is clear and useful to the user.

SUP.1.4 Distribute document. Package and distribute the document as paper, electronic, or other media to the appropriate parties.

SUP.1.5 Maintain document. Maintain the document, and when it becomes necessary to modify it, perform the previous activities as appropriate.

Note: 1) If the document is part of a product baseline or if its control and stability are important, it should be modified and distributed in accordance with process, "Perform configuration management" SUP.2.

2) If the document is part of a product baseline under maintenance, its maintenance is covered by process ENG.7.

SUP.2 Perform configuration management

The purpose of the *Perform configuration management* process is to establish and maintain the integrity of all of the products of the software project throughout the project's software life cycle.

Note: Generic practice 2.2.2 at the Planned-and-Tracked Level instantiates this process for selected work products.

SUP.2.1 Establish configuration management library system. Establish and manage a repository with access controls that provides for

- storage and retrieval of configuration items (and their versions);
- sharing and transfer of configuration items between affected groups;
- recovery of archive versions of configuration items;
- correct creation of products from the library.

SUP.2.2 Identify configuration items. Identify each work product to be placed under configuration management.

Note: Examples include

- requirements, designs, code, tests;
- other product baselines (e.g., user documentation);
- software project plans;
- standards and procedures.

SUP.2.3 Maintain configuration item descriptions. Provide and maintain a description of each configuration item, identifying

- its decomposition into lower level configuration components;
- the person responsible for each item;
- when placed under configuration management.

SUP.2.4 Manage change requests. Record, review, approve, and track all change requests and problem reports for all configuration items and their versions.

SUP.2.5 Control changes. Provide access control to help maintain the correctness and integrity of the software items in the configuration management library system.

SUP.2.6 Build product releases. Build product releases only from configuration items in the library and only when authorized.

SUP.2.7 Maintain configuration item history. Maintain a history of each configuration item, recording configuration management actions against the item in sufficient detail to allow for recovery of previous versions.

SUP.2.8 Report configuration status. Regularly report on the results of performing the above activities and the status of each configuration item.

SUP.3 Perform quality assurance

The purpose of the *Perform quality assurance* process is to ensure that work products and activities comply with all applicable standards, procedures, and requirements.

The key requirement here is that an objective view of the quality of the process and work products be determined and reported. Quality Assurance can be implemented in different ways; it does not need to be performed by a separate group (e.g. a group called the "Software Quality Assurance Group").

Note: There is another process with a similar name, "Manage quality" PRO.5. Process PRO.5 focuses on identifying what needs to be done to build quality into the products and establishing management controls to ensure this gets done; whereas SUP.3 focuses more on an audit and review approach and on ensuring compliance.

SUP.3.1 Select project standards. Contribute to the project's software plans, by evaluating completeness of the plans and helping select the standards and procedures that will be used on the project.

SUP.3.2 Review software engineering activities. Review the software engineering activities against the plans and the selected standards and procedures.

SUP.3.3 Audit work products. Audit software work products against the selected standards and procedures.

SUP.3.4 Report results. Report the results of the above activities, in particular, deviations, to the appropriate levels of management and staff.

Note: The above is addressed by generic practices 2.3.1 and 2.3.2 at the Planned-and-Tracked Level.

SUP.3.5 Handle deviations. Deviations are addressed at the appropriate level of management, going to the next higher level, where necessary, until resolved.

SUP.4 Perform problem resolution

The purpose of the *Perform problem resolution* process is to ensure that all discovered problems are analyzed and removed, and trends are identified.

The problems being addressed here are any detected problems whatever their nature or source. Example sources are base practices

- CUS.4.3-4.5 (problems identified in customer audits and reviews);
- CUS.6.4 (problems relating to operation of software);
- CUS.6.5 (problems relating to use of software);
- CUS.7.3 (problems identified by monitoring performance);
- ENG.5.4, 5.6 (problems identified during software testing);
- ENG 6.3, 6.5 (problems identified during system testing);
- ENG.7.2 (user problems identified during maintenance);
- SUP.3.5 (deviations from requirements, standards, or procedures);
- SUP.5.7 (problems identified during peer reviews).

There are many other sources among the base practices, for example, those involving taking corrective action.

SUP.4.1 Prepare problem report. Prepare a problem report promptly after each problem is detected describing the nature of the problem.

Note: It may be the user or customer who does this, in which case this base practice can be considered, "not applicable."

SUP.4.2 Track problem report. Track the resolution of the problem/change report.

SUP.4.3 Prioritize problems. Categorize and prioritize according to the priority and category of the problem.

SUP.4.4 Determine resolution. Analyze the problem and, if possible, determine the problem's cause and document its resolution.

Note: The order in which problems are resolved may be prioritized by severity or underlying trends.

SUP.4.5 Correct the defect. Eliminate the defect in the product.

SUP.4.6 Distribute the correction. Distribute the corrected product.

Note: Each release of the product will normally include a number of defect corrections.

SUP.5 Perform peer reviews

The purpose of the *Perform peer reviews* process is to efficiently find and remove defects from products produced by the project.

When reviews involve the customer or management, usually at the end of a task, they are called, "technical reviews." For important work products (e.g., requirements, designs, code), it is important to hold a review early in the task so that defects can be efficiently found and rework significantly reduced. In such reviews, the emphasis is on technical correctness, and so the reviewers are typically the colleagues of the work product's producer (as opposed to management or the customer). Such reviews are called, "peer reviews."

There are different methods for carrying out a peer review by one's peers. Some of these go by the name, "inspections." This process description does not favour a particular method, only that there be (at least) one.

The work products to be peer reviewed are identified in the project's software plans (see process, "Establish project plan" PRO.2).

The action items which are output by this process are input to "Perform problem resolution" SUP.4 process.

Note: Generic practice 3.2.2 at the Well-Defined Level instantiates this process for selected work products. For example, in process, "Develop software design" ENG.3, an important work product is the software architecture (see base practice ENG.3.1). When assessing process ENG.3 against generic practice 3.2.2, we might ask whether software architectures are peer reviewed. The process we would expect to see applied when peer reviewing the software architecture, would correspond, at a very high level of description, with what is described here for process SUP.5.

SUP.5.1 Select work products. Identify the work products that are to undergo peer review.

Note: This is one aspect of planning the peer reviews, which is covered by a generic practice.

SUP.5.2 Identify review standards. Identify the standards (including checklists) to be used in conducting the peer reviews.

SUP.5.3 Establish completion criteria. Establish the completion criteria for successful completion of peer reviews.

SUP.5.4 Establish re-review criteria. Establish criteria for when and how to re-review a work product.

SUP.5.5 Distribute review materials. Distribute the materials for peer reviews well in advance of the reviews.

SUP.5.6 Conduct peer review. Conduct peer review on the selected work product.

SUP.5.7 Document action items. Document action items identified during peer reviews.

SUP.5.8 Track action items. Track to closure action items identified during peer reviews.

6.5 Organization process category (ORG)

The *Organization* process category consists of processes which establish the business goals of the organization and develop process, product, and resource assets which, when used by the projects in the organization, will help the organization achieve its business goals.

These organizational processes:

- build organizational infrastructure;
- leverage off the best of what is available in any one part of the organization (effective processes, advanced skills, quality code, good support tools);
- make it available to all;

When applying the common features to organization-level processes, note that the defined process described in the generic practices may be implemented at the organization as well as project level.

The processes belonging to the "Organization" process category are:

ORG.1 Engineer the business

ORG.2 Define the process

ORG.3 Improve the process

ORG.4 Perform training

ORG.5 Enable reuse

ORG.6 Provide software engineering environment

ORG.7 Provide work facilities

ORG.1 Engineer the business

The purpose of the *Engineer the business* process is to provide the individuals in the organization and projects with a vision and culture which empowers them to function effectively.

Although business re-engineering and Total Quality Management have a much broader scope than that of software process, software process improvement occurs in a business context and, to be successful, must address business goals.

ORG.1.1 Establish strategic vision. Establish a strategic vision for the organization that identifies what business the (software producing part of) the organization is in.

ORG.1.2 Deploy vision. Deploy the organization's strategic vision to all individuals working for the organization.

ORG.1.3 Establish quality culture. Establish an organizational culture which supports a Total Quality focus on customer satisfaction.

ORG.1.4 Build integrated teams. Build teams with an integrated product perspective whose goal is to satisfy the customer.

ORG.1.5 Provide incentives. Provide incentives to team members to work as a team to accomplish the team's goal.

Note: Incentives may be provided by:

- establishing joint responsibility for team performance from problem formulation (beginning) to solution implementation (end);
- including peer inputs in performance reviews;
- linking rewards and recognition to team performance.

ORG.1.6 Define career plans. Define career plans for the individuals in the organization.

Note: In the case of small organizations with limited growth potential, this may simply take the form of an employee development plan (a plan for improving the knowledge, skills, and experience base of the employee).

ORG.2 Define the process

The purpose of the *Define the process* process is to build a reusable library of process definitions (including standards, procedures, and models) that will support stable and repeatable performance of the software engineering and management process (all the processes covered in this guide).

This process is an integral part of the common feature "Defining a Standard Process" at the Well-Defined Level. At least partial performance of this process is therefore a prerequisite for any other process achieving the Well-Defined Level, although it need not achieve the Well-Defined Level itself.

ORG.2.1 Define goals. Define the process goals that are to be achieved by following the process.

Note: One input to defining the goals is the organization's strategic vision. When defining a goal, it is important to state it in a manner so that its achievement can be measured and determined with some objectivity. See base practice, "Define process measures" ORG.2.9.

ORG.2.2 Identify current activities, roles & responsibilities. Identify the activities that comprise the way the process is currently and/or should be performed and identify the roles and responsibilities for these activities.

ORG.2.3 Identify inputs and outputs. Identify the inputs and outputs for the process.

ORG.2.4 Define entry and exit criteria. Define the criteria for entering and exiting the process.

ORG.2.5 Define control points. Define the points in the process where key reviews and decisions are made.

ORG.2.6 Identify external interfaces. Identify the interfaces with related processes, which supply inputs and consume outputs.

Note: Base practice, "Identify inputs and outputs" ORG.2.3 identifies the work products entering and leaving the process. This base practice identifies the relationship to other processes that this process may need to work with (e.g. management processes as well as supplier/consumer processes) and inputs and outputs between them.

ORG.2.7 Identify internal interfaces. Identify the interfaces between the activities in the process.

ORG.2.8 Define quality records. Define the quality records that will demonstrate conformance to the process and determine their retention period..

ORG.2.9 Define process measures. Define measures for the process that can be used to determine achievement of the process goals.

Note. These measures will address such goals as process efficiency and quality.

ORG.2.10 Document the standard process. Document the standards, procedures, and models for performing the process and characterizing its outputs.

Note: Example areas that might be covered by software engineering standards include

- requirements specification;
- design methods;
- coding style;
- programming languages;
- testing;
- security;
- human factors;
- documentation;
- project management plans;
- software quality assurance plans;
- configuration management plans.

ORG.2.11 Establish policy. Establish a written organization policy for performing the process.

ORG.2.12 Establish performance expectations. Establish expectations for process performance when using the organization's standard process family.

Note: These expectations will typically be quantitative. The policy in ORG.2.11 sets expectations for what process will be performed and how; this practice quantifies the expectations for performance. Even when these expectations are not be well-bounded in the sense of a Level 4 capability, they may include quantitative criteria that guide effective implementation of the process.

ORG.2.13 Deploy the process. Deploy the organization's standard process family available throughout the organization.

Note: Deployment of the organization’s standard process family will frequently involve training.

ORG.3 Improve the process

The purpose of the *Improve the process* process is to continually improve the effectiveness and efficiency of the processes used by the organization in line with the busines need.

Note: The focus of this process will typically be the standard family of processes, which are built according to generic practice 3.1.1. That is why this process is considered an organizational process: organizational standards and process assets provide a common basis for continual process improvement. Although there will not usually be sufficient time to go through a full process improvement cycle at the project level, this process can also be used to improve the processes of large, long-duration projects.

ORG.3.1 Identify improvement opportunities. Identify opportunities for software process improvement by regularly analyzing

- measures of software quality and productivity, possible process and technology changes;
- internal and external comparisons (benchmarks);
- contract and product requirement changes.

ORG.3.2 Define scope of improvement activities. Define the purpose, objectives, scope, and priorities of the process improvement activities in accordance with the business goals of the organization.

ORG.3.3 Understand the process. Assess the process to understand its strengths and weaknesses

ORG.3.4 Identify improvements. Identify where the process needs to be improved to achieve its process goals.

Note: See base practice, "Define goals" ORG.2.1, for more on process goals.

ORG.3.5 Prioritize improvements. Prioritize the improvements which can be made in the process based on an analysis of the impact of potential improvements on achieving the goals of the process.

ORG.3.6 Define measures of impact. Define measures that can be used to determine the impact of the process changes on achieving the process's goals.

Note: These measures would include those identified in base practice, "Define process measures" ORG.2.9.

ORG.3.7 Change the process. Change the process to improve it.

ORG.3.8 Confirm the improvement. Pilot test changes to confirm that they improve the process based on analysis of appropriate data.

ORG.3.9 Deploy improvement. Deploy improved processes across the organization as appropriate.

ORG.4 Perform training

The purpose of the *Perform training* process is to provide the organization and projects with individuals who possess the needed skills and knowledge to perform their roles effectively.

Training encompasses more than classroom hours of instructor-led education. Other activities also serve to satisfy the purpose of this process such as on-the-job training and mentoring.

Note: "Perform training" supports the generic training practice 2.1.5 in "Planning performance" at the Planned-and-Tracked Level, so organization-level training is not required to satisfy that practice.

ORG.4.1 Identify training needs. Identify common training needs across the organization based on organizational and project inputs to build the knowledge and skills of the staff.

ORG.4.2 Develop or acquire training. Develop or acquire training that addresses the common training needs.

ORG.4.3 Train personnel. Train personnel to have the knowledge and skills needed to perform their roles.

ORG.4.4 Maintain training records. Maintain appropriate records of training and experience for the staff.

ORG.5 Enable reuse

The purpose of the *Enable reuse* process is to maximize reuse of existing system and software components, leading to lower development and maintenance costs and higher quality product lines.

ORG.5.1 Determine organizational reuse strategy. Determine which product lines and types of work products should be supported with reuse.

Note: Much reuse can be accomplished within product lines. When software architectures are fairly constant across projects, high levels of design reuse are possible.

ORG.5.2 Identify reusable components. Identify system and software components which could profitably be reused within the product lines established by the organization.

Note: Consider only high quality system and software components for reuse.

ORG.5.3 Develop reusable components. Develop system and software components which are designed for reuse.

ORG.5.4 Establish a reuse library. Establish a library of reusable components, including the mechanisms for identifying and retrieving components.

ORG.5.5 Certify reusable components. Certify that components placed in the reuse library have been appropriately packaged for reuse.

ORG.5.6 Integrate reuse into life cycle. Modify the software life cycle and/or standard process to address the incorporation of reusable components as appropriate.

ORG.5.7 Propagate change carefully. Before making a global change to a reusable component, evaluate the impact of the change, and the impact to systems and software in which it was incorporated.

Note: Sometimes due to defects, improvements in algorithms, etc., it is desirable to propagate a change in a reusable component to many of the systems and software in which it was incorporated. Before doing so, carefully evaluate the impact of propagating the change.

ORG.6 Provide software engineering environment

The purpose of the *Provide software engineering environment* process is to provide an integrated set of software development tools for use by the projects in the organization, consistent and supportive of the process standard.

ORG.6.1 Identify software engineering environment requirements. Determine requirements for the software engineering environment, identifying

- process roles and activities it should support;
- security issues it should address;
- throughput and data sharing requirements;
- backup and recovery.

ORG.6.2 Provide a software engineering environment. Acquire and provide a software engineering environment which satisfies the requirements.

ORG.6.3 Provide support for developers. Provide support for the developers who will utilize the software engineering environment.

Note: Support includes identifying and resolving problems arising from the use of provided tools and facilities.

ORG.6.4 Maintain software engineering environment. Perform maintenance on the software engineering environment for the purposes of

- correcting defects;
- improving performance;
- modifying the environment to keep up with changes in the process activities and tools it supports;
- controlling changes to enable regression if necessary.

ORG.7 Provide work facilities

The purpose of the *Provide work facilities* process is to provide a secure and reliable environment in which software project activities may be carried out.

ORG.7.1 Provide productive workspace. Provide a productive workspace, with appropriate furnishings and office equipment.

Note: Possible attributes of a productive workspace include: safety, quietness, and comfort.

ORG.7.2 Ensure data integrity. Provide the means to ensure that data resulting from the software project's activities are appropriately archived and protected from corruption.

ORG.7.3 Provide data backups. Provide the means for performing regular backup and archival of data generated by the software project's activities to prevent loss.

ORG.7.4 Provide building facilities. Provide supportive building facilities which may include

- access to meeting rooms;
- adequate storage space;
- access to parking;
- access to rest rooms;
- adequate communication facilities;
- adequate lighting;
- adequate environmental control (e.g. heating/cooling);
- adequate refreshments.

ORG.7.5 Provide remote access facility. Provide the software project's technical and managerial staff with the means to access their work environment and data from a remote location, as appropriate.

Annex A (normative)

Extended processes and variant models

A.1 Requirements for extended process and variant models

In order to address the unique needs of a specific organisation or industry sector, other models may be built by selecting specific processes from the model defined in this part of the International Standard and providing guidance on how to interpret the practices. Alternatively, processes may be extended by including additional base practices, possibly supplemented by guidance on how to interpret the adequacy of practices. An extended process may also be an entirely new process. These models, referred to as variant models, shall satisfy the following criteria.

A variant model shall be a proper (non-empty) subset of the processes defined in this part of the International Standard. Model practices should be separately identifiable from any tailoring that may be made for purposes of comparison across conformant variants.

An extended process shall not change or delete base practices.

New processes may be added to a variant model, but these new processes shall not contain base practices drawn from other processes within this part of the International Standard. This means that it is not possible to construct a variant model by selecting a subset of base practices from a number of processes and then deleting all of those processes.

A variant model shall be documented to identify its differences from this part of the International Standard.

A variant model shall provide direct traceability between its base practices and the base practices in this part of the International Standard.

Extended processes shall be documented according to the nomenclature defined in clause 4.

A.2 Guidelines for the construction of extended processes

Extended processes should be documented according to the style guide in Annex H.

New processes should have a defined purpose.

New processes should have a sufficient set of base practices to permit the transformation of its inputs into its outputs and to meet the defined purpose of the process.

The common features and generic practices in this part of the International Standard should be applicable to extended processes.

Extended processes may provide guidance on the appropriate interpretation of a practice. This may be implemented as adequacy indicators to be incorporated into an assessment instrument as described in part 6 of this International Standard.

Extended processes may add new base practices to those in its equivalent in this part of the International Standard.

The base practices in an extended process should be checked for completeness.

The base practices in an extended process should be checked to ensure they are necessary for achieving the purpose of the process.

Annex B (informative)

Summary list of practices

B.1 Summary list of base practices

As in clause 6, the base practices are grouped under the process they belong to and processes are grouped under the process category they belong to. Titles only are shown. This summary provides a high-level overview of the primary activities the model prescribes for each process. It can be used to get a quick understanding of the content of this part of the International Standard, except for the generic practices. This annex is for informational purposes only - it is not intended to replace clause 6 in assessments.

CUS Customer-Supplier process category

CUS.1 Acquire software product and/or service

CUS.1.1 Identify the need
CUS.1.2 Define the requirements
CUS.1.3 Prepare acquisition strategy
CUS.1.4 Prepare request for proposal
CUS.1.5 Select software product supplier

CUS.2 Establish contract

CUS.2.1 Review before contract finalization
CUS.2.2 Negotiate contract
CUS.2.3 Determine interfaces to independent agents
CUS.2.4 Determine interfaces to subcontractors

CUS.3 Identify customer needs

CUS.3.1 Obtain customer requirements and requests
CUS.3.2 Understand customer expectations
CUS.3.3 Keep customers informed

CUS.4 Perform joint audits and reviews

CUS.4.1 Establish joint audits and reviews
CUS.4.2 Prepare for customer audits and reviews
CUS.4.3 Conduct joint management reviews
CUS.4.4 Conduct joint technical reviews
CUS.4.5 Support customer acceptance review
CUS.4.6 Perform joint process assessment

CUS.5 Package, deliver, and install the software

CUS.5.1 Identify installation requirements
CUS.5.2 Prepare site for installation
CUS.5.3 Pack software
CUS.5.4 Deliver software
CUS.5.5 Verify correct receipt
CUS.5.6 Install software
CUS.5.7 Provide handling and storage procedures

CUS.6 Support operation of software

CUS.6.1 Identify operational risks
CUS.6.2 Perform operational testing
CUS.6.3 Operate the software
CUS.6.4 Resolve operational problems
CUS.6.5 Handle user requests
CUS.6.6 Document temporary work-arounds
CUS.6.7 Monitor system capacity and service

CUS.7 Provide customer service

CUS.7.1 Train customer
CUS.7.2 Establish product support
CUS.7.3 Monitor performance
CUS.7.4 Install product upgrades

CUS.8 Assess customer satisfaction

CUS.8.1 Determine customer satisfaction level
CUS.8.2 Compare with competitors
CUS.8.3 Communicate customer satisfaction

ENG Engineering process category

ENG.1 Develop system requirements and design

ENG.1.1 Specify system requirements
ENG.1.2 Describe system architecture
ENG.1.3 Allocate requirements
ENG.1.4 Determine release strategy

ENG.2 Develop software requirements

ENG.2.1 Determine software requirements
ENG.2.2 Analyze software requirements
ENG.2.3 Determine operating environment impact
ENG.2.4 Evaluate requirements with customer
ENG.2.5 Update requirements for next iteration

ENG.3 Develop software design

ENG.3.1 Develop software architectural design
ENG.3.2 Design interfaces at top level
ENG.3.3 Develop detailed design
ENG.3.4 Establish traceability

ENG.4 Implement software design

ENG.4.1 Develop software units
ENG.4.2 Develop unit verification procedures
ENG.4.3 Verify the software units

ENG.5 Integrate and test software

ENG.5.1 Determine regression test strategy
ENG.5.2 Build aggregates of software units
ENG.5.3 Develop tests for aggregates
ENG.5.4 Test software aggregates
ENG.5.5 Develop tests for software
ENG.5.6 Test integrated software

ENG.6 Integrate and test system

ENG.6.1 Build aggregates of system elements
ENG.6.2 Develop tests for aggregates
ENG.6.3 Test system aggregates
ENG.6.4 Develop tests for system
ENG.6.5 Test integrated system

ENG.7 Maintain system and software

ENG.7.1 Determine maintenance requirements
ENG.7.2 Analyze user problem and enhancements
ENG.7.3 Determine modifications for next upgrade
ENG.7.4 Implement and test modifications
ENG.7.5 Upgrade user system

PRO Project process category

PRO.1 Plan project life cycle

PRO.1.1 Evaluate options for product development
PRO.1.2 Select software life cycle model
PRO.1.3 Describe activities and tasks
PRO.1.4 Establish task sequences
PRO.1.5 Document activities

PRO.2 Establish project plan

PRO.2.1 Develop work breakdown structure
PRO.2.2 Identify project standards
PRO.2.3 Identify specialized facilities
PRO.2.4 Determine reuse strategy
PRO.2.5 Develop project estimates
PRO.2.6 Identify initial project risks
PRO.2.7 Identify project measures
PRO.2.8 Establish project schedule
PRO.2.9 Establish project commitments
PRO.2.10 Document project plans

PRO.3 Build project teams

PRO.3.1 Define project teams
PRO.3.2 Empower project teams
PRO.3.3 Maintain project team interactions
PRO.3.4 Manage inter-team issues

PRO.4 Manage requirements

PRO.4.1 Agree on requirements
PRO.4.2 Establish customer requirements baseline
PRO.4.3 Manage customer requirements changes
PRO.4.4 Use customer requirements
PRO.4.5 Maintain traceability

PRO.5 Manage quality

PRO.5.1 Establish quality goals
PRO.5.2 Define quality metrics
PRO.5.3 Identify quality activities
PRO.5.4 Perform quality activities
PRO.5.5 Assess quality
PRO.5.6 Take corrective action

PRO.6 Manage risks

PRO.6.1 Establish risk management scope
PRO.6.2 Identify risks
PRO.6.3 Analyze and prioritize risks
PRO.6.4 Develop mitigation strategies
PRO.6.5 Define risk metrics
PRO.6.6 Implement mitigation strategies
PRO.6.7 Assess results of mitigation strategies
PRO.6.8 Take corrective action

PRO.7 Manage resources and schedule

PRO.7.1 Acquire resources
PRO.7.2 Track progress
PRO.7.3 Conduct management reviews
PRO.7.4 Conduct technical reviews
PRO.7.5 Manage commitments

PRO.8 Manage subcontractors

PRO.8.1 Establish statement of work
PRO.8.2 Qualify potential subcontractors
PRO.8.3 Select subcontractor
PRO.8.4 Establish and manage commitments
PRO.8.5 Maintain communications
PRO.8.6 Assess compliance
PRO.8.7 Assess subcontractor quality

SUP Support process category

SUP.1 Develop documentation

SUP.1.1 Determine documentation requirements
SUP.1.2 Develop document
SUP.1.3 Check document
SUP.1.4 Distribute document
SUP.1.5 Maintain document

SUP.2 Perform configuration management

SUP.2.1 Establish configuration management library system
SUP.2.2 Identify configuration items
SUP.2.3 Maintain configuration item descriptions
SUP.2.4 Manage change requests
SUP.2.5 Control changes
SUP.2.6 Build product releases
SUP.2.7 Maintain configuration item history
SUP.2.8 Report configuration status

SUP.3 Perform quality assurance

SUP.3.1 Select project standards
SUP.3.2 Review software engineering activities
SUP.3.3 Audit work products
SUP.3.4 Report results
SUP.3.5 Handle deviations

SUP.4 Perform problem resolution

SUP.4.1 Prepare problem report
SUP.4.2 Track problem report
SUP.4.3 Prioritize problems
SUP.4.4 Determine resolution
SUP.4.5 Correct the defect
SUP.4.6 Distribute the correction

SUP.5 Perform peer reviews

SUP.5.1 Select work products
SUP.5.2 Identify review standards
SUP.5.3 Establish completion criteria
SUP.5.4 Establish re-review criteria
SUP.5.5 Distribute review materials
SUP.5.6 Conduct peer review
SUP.5.7 Document action items
SUP.5.8 Track action items

ORG Organization process category

ORG.1 Engineer the business

ORG.1.1 Establish strategic vision
ORG.1.2 Deploy vision
ORG.1.3 Establish quality culture
ORG.1.4 Build integrated teams
ORG.1.5 Provide incentives
ORG.1.6 Define career plans

ORG.2 Define the process

ORG.2.1 Define goals
ORG.2.2 Identify current activities, roles and responsibilities
ORG.2.3 Identify inputs and outputs
ORG.2.4 Define entry and exit criteria
ORG.2.5 Define control points
ORG.2.6 Identify external interfaces
ORG.2.7 Identify internal interfaces
ORG.2.8 Define quality records
ORG.2.9 Define process measures
ORG.2.10 Document the standard process
ORG.2.11 Establish policy
ORG.2.12 Establish performance expectations
ORG.2.13 Deploy the process

ORG.3 Improve the process

ORG.3.1 Identify improvement opportunities
ORG.3.2 Define scope of improvement activities
ORG.3.3 Understand the process
ORG.3.4 Identify improvements
ORG.3.5 Prioritize improvements
ORG.3.6 Define measures of impact
ORG.3.7 Change the process
ORG.3.8 Confirm the improvement
ORG.3.9 Deploy improvement

ORG.4 Perform training

ORG.4.1 Identify training needs
ORG.4.2 Develop or acquire training
ORG.4.3 Train personnel
ORG.4.4 Maintain training records

ORG.5 Enable reuse

ORG.5.1 Determine organizational reuse strategy
ORG.5.2 Identify reusable components
ORG.5.3 Develop reusable components
ORG.5.4 Establish a reuse library
ORG.5.5 Certify reusable components
ORG.5.6 Integrate reuse into life cycle
ORG.5.7 Propagate change carefully

ORG.6 Provide software engineering environment

ORG.6.1 Identify software engineering environment requirements
ORG.6.2 Provide a software engineering environment
ORG.6.3 Provide support for developers
ORG.6.4 Maintain software engineering environment

ORG.7 Provide work facilities

ORG.7.1 Provide productive workspace
ORG.7.2 Ensure data integrity
ORG.7.3 Provide data backups
ORG.7.4 Provide building facilities
ORG.7.5 Provide remote access facility

B.2 Summary list of generic practices

This summary provides a high-level overview of the capability levels in the model. As in clause 5, the generic practices are grouped under common features and capability levels. Titles only are shown. This annex is for informational purposes only - it is not intended to replace Section 5 in assessments.

Level 1: Performed-Informally Level

Common Feature 1.1: Performing Base Practices

1.1.1 Perform the process.

Level 2: Planned-and-Tracked Level

Common Feature 2.1: Planning Performance

2.1.1 Allocate resources.
2.1.2 Assign responsibilities.
2.1.3 Document the process.
2.1.4 Provide tools.
2.1.5 Ensure training.
2.1.6 Plan the process.

Common Feature 2.2: Disciplined Performance

2.2.1 Use plans, standards, and procedures.
2.2.2 Do configuration management.

Common Feature 2.3: Verifying Performance

2.3.1 Verify process compliance.
2.3.2 Audit work products.

Common Feature 2.4: Tracking Performance

2.4.1 Track with measurement.
2.4.2 Take corrective action.

Level 3: Well-Defined Level

Common Feature 3.1: Defining a Standard Process

3.1.1 Standardize the process.
3.1.2 Tailor the standard process.

Common Feature 3.2: Performing the Defined Process

3.2.1 Use a well-defined process.
3.2.2 Perform peer reviews.
3.2.3 Use well-defined data.

Level 4: Quantitatively-Controlled Level

Common Feature 4.1: Establishing Measurable Quality Goals

4.1.1 Establish quality goals.

Common Feature 4.2: Objectively Managing Performance

4.2.1 Determine process capability.
4.2.2 Use process capability.

Level 5: Continuously-Improving Level

Common Feature 5.1: Improving Organizational Capability

5.1.1 Establish process effectiveness goals.
5.1.2 Continuously improve the standard process.

Common Feature 5.2: Improving Process Effectiveness

5.2.1 Perform causal analysis.
5.2.2 Eliminate defect causes.
5.2.3 Continuously improve the defined process.

Annex C (Informative)

Components of the model

C.1 Introduction

The process model in this part of the International Standard defines processes and practices that may be implemented to establish and improve an organization's software acquisition, development, maintenance, operation and support capabilities. Practices in this model are organized using an architecture that will help software personnel understand how to continuously improve the management of software processes. The architecture has been designed to facilitate the assessment of an organization's software processes and make judgements and recommendations regarding improvements to them.

Each process in the model is described by base practices, which are the essential activities of that specific process. Processes, in turn, are grouped into five process categories.

Evolving process capability is expressed in terms of capability levels, common features, and generic practices. Generic practices implement or institutionalize a process in a general way. These practices represent the activities necessary to manage a process and improve its capability to perform. They are potentially applicable to any process. They are grouped into common features and capability levels according to the aspect of implementation or institutionalization they address.

C.2 Definitions of architecture components

C.2.1 Grouping by type of activity

The first set of architectural components — process category, process, base practices — is a grouping by *type of activity*.

Process Category Each process category is a set of processes addressing the same general area of activity.

The process categories covered in this part of the International Standard address five general areas of activity: customer-supplier, engineering, project, support, and organization.

The **Customer-Supplier** process category consists of processes that directly impact the customer, supporting development and transition of the software to the customer, and provide for its correct operation and use.

The **Engineering** process category consists of processes that directly specify, implement, or maintain a system and software product and its user documentation.

The **Project** process category consists of processes which establish the project, and co-ordinate and manage its resources to produce a product or provide services which satisfy the customer.

The **Support** process category consists of processes which enable and support the performance of the other processes on a project.

The **Organization** process category consists of processes which establish the business goals of the organization and develop process, product, and resource assets which will help the organization achieve its business goals.

These process categories are related as illustrated in figure 1. "Customer-Supplier" process category consists of the processes closest to the customer. We can think of each succeeding process category as one layer removed from the preceding layer. Thus the "Engineering" process category consists of processes which build the product which is delivered to the customer. The "Project" process category consists of processes which manage the development of the product. The "Support" process category consists of processes which enable and support the performance of the previous processes, whether management, engineering, or customer-related. Finally, the "Organizational" process category consists of processes that build organizational infrastructure on which all previous processes can potentially exercise the most benefit to the organization.

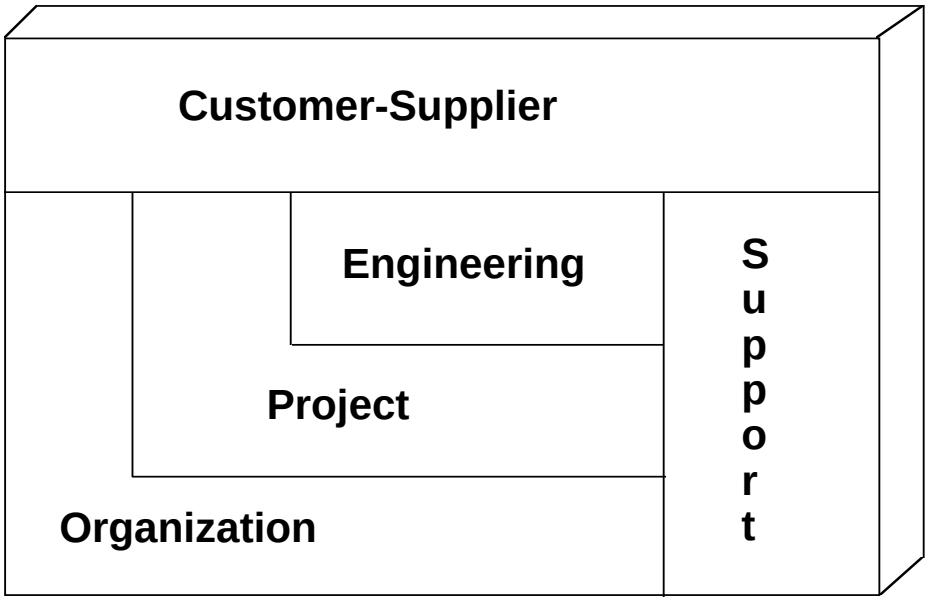

Figure 1 – Interrelationship of process categories.

Process A process is a set of activities that achieves a purpose. Each process in this part of the International Standard has a purpose and consists of a set of practices that address that purpose.

For example, the Engineering process "Develop software design" ENG.3 contains this purpose statement: "The purpose of the *Develop software design* process is to establish a software design that effectively accommodates the software requirements; at the top-level this identifies the major software components and refines these into lower level software units which can be coded, compiled, and tested." The purpose statement indicates the reason for performing the "Develop software design" ENG.3 process.

The processes in this part of the International Standard are not processes in the sense of being complete process models or descriptions. These processes contain essential practices, but they do not describe how to perform the process. This part of the International Standard is a descriptive model, not a prescriptive model.

A summary list of all the processes in this model are listed in Annex B.

Base Practice A base practice is a software engineering or management activity that addresses the purpose of a particular process. Consistently performing the base practices associated with a process will help in consistently achieving its purpose.

Thus a process consists of a set of *base practices.* For example, "Develop software design" ENG.3 consists of the following set of base practices:

ENG.3.1 Develop software architectural design

ENG.3.2 Design interfaces at top level

ENG.3.3 Develop detailed design

ENG.3.4 Establish traceability

Note that these practices should be performed to achieve the defined purpose of the process.

The base practices in this part of the International Standard are described at an abstract level, identifying "what" should be done without specifying "how". See other ISO JTC1/SC7 standards for further information and guidance on implementing these practices.

These processes and activities characterize performance of a process, even if that performance is not systematic. Performance of the base practices may be ad hoc, unpredictable, inconsistent, poorly planned, and/or result in poor quality products, but those work products are at least marginally usable in achieving the purpose of the process. Implementing only the base practices of a process may be of minimal value and represents only the first step in building process capability, but the base practices represent the unique, functional activities of the process.

C.2.2 Grouping by type of implementation or institutionalization activity

The second set of architectural components - capability levels, common features, and generic practices - groups by *type of implementation or institutionalization activity*. They deal, at different levels of abstraction, with the activities necessary to manage a process and improve its capability to perform.

Capability Level A capability level is a set of common features (sets of activities) that work together to provide a major enhancement in the capability to perform a process.

For example, to effectively attain conformance to the practices at the Well-Defined Level (Level 3), for some processes the organization will require support through an effective implementation of another process, namely "Define the process" ORG.2.

Capability levels provide two benefits: they acknowledge dependencies among the practices of a process, and they help an organization identify which improvements it might perform first, based on a rational sequence of process implementation. Capability levels can be represented either by number or by title, but it is recommended that the titles be generally used instead of the numbers as they are more descriptive of the practices the levels contain.

Each level provides a major enhancement in capability to that provided by its predecessors in the performance of a process. Together, they constitute a route map for improving a specific process in a logical fashion. The capability levels are a rational way of progressing through enhancements to a process. This may not, however, apply to all software environments. Organization-wide improvement considerations and taking advantage of what is already implemented in an organization may influence progression through the capability levels. It should also be noted that there are dependencies between processes, and the successful satisfaction of a capability level within a process may require support from another process.

Capability levels

0	**Not-Performed**	The Not-Performed level has no common features. There is general failure to perform the base practices in the process. There are no easily identifiable work products or outputs of the process.
1	**Performed-Informally**	Base practices of the process are generally performed. The performance of these base practices may not be rigorously planned and tracked. Performance depends on individual knowledge and effort. Work products of the process testify to the performance. Individuals within the organization recognize that an action should be performed, and there is general agreement that this action is performed as and when required. There are identifiable work products for the process.
2	**Planned-and-Tracked**	Performance of the base practices in the process is planned and tracked. Performance according to specified procedures is verified. Work products conform to specified standards and requirements. The primary distinction from the Performed-Informally Level is that the performance of the process is planned and managed and progressing towards a well-defined process
3	**Well-Defined**	Base practices are performed according to a well-defined process using approved, tailored versions of standard, documented processes. The primary distinction from the Planned-and-Tracked Level is that the process of the Well-Defined Level is planned and managed using an organization-wide standard process.
4	**Quantitatively-Controlled**	Detailed measures of performance are collected and analyzed. This leads to a quantitative understanding of process capability and an improved ability to predict performance. Performance is objectively managed. The quality of work products is quantitatively known. The primary distinction from the Well-Defined Level is that the defined process is quantitatively understood and controlled.
5	**Continuously-Improving**	Quantitative process effectiveness and efficiency goals (targets) for performance are established, based on the business goals of the organization. Continuous process improvement against these goals is enabled by quantitative feedback from performing the defined processes and from piloting innovative ideas and technologies. The primary distinction from the Quantitatively-Controlled Level is that the defined process and the standard process undergo continuous refinement and improvement, based on a quantitative understanding of the impact of changes to these processes.

Common Feature A common feature is a set of practices that address an aspect of process implementation or institutionalization.

As an example, the Planned-and-Tracked Level contains the Planning Performance, Disciplined Performance, Verifying Performance, and Tracking Performance common features (See clause 5). The Tracking Performance common feature consists of practices that track process status using measurement and take corrective action as appropriate.

The common features associated with capability levels 1-5 are given below. The names of the common features are italicized. Under each is a bulleted short description of the kinds of things that need to be addressed by the organization in order to implement the common feature. (The generic practices belonging to each common feature specifically address these bullets.)

Performed-Informally Level

Performing Base Practices

- perform the process

Planned-and-Tracked Level

When interpreting the common features for the Planned-and-Tracked Level, remember that the particular place in the organization where these practices are performed is not specified. It can be at the level of an individual group, a project, or the entire organization. In most instances, the following are expected to be performed at the level of a project.

Planning Performance

- allocate resources
- assign responsibilities
- document the process
- provide tools
- ensure training
- plan the process

Disciplined Performance

- use plans, standards, and procedures
- do configuration management

Verifying Performance

- verify process compliance
- audit work products

Tracking Performance

- track with measurement
- take corrective action

Well-Defined Level

When interpreting the common features for the Well-Defined Level, remember that: 1) there can be more than one "standard" process for the same process in an organization (e.g. due to developing software for different applications or markets), and 2) the place in the organization where tailoring and use of the defined process takes place is not specified, though it will normally be at the level of the project. For example, organization processes are typically not performed by software projects.

Defining a Standard Process

- standardize the process
- tailor the standard process

Performing the Defined Process

- use a well-defined process
- perform peer reviews
- use well-defined data

Quantitatively-Controlled Level

When interpreting the common features for the Quantitatively-Controlled Level, note that each defined process is a tailored version of the standard process, and that each standard process covers the base practices for that process.

Establishing Measurable Quality Goals

- establish quality goals

Objectively Managing Performance

- determine process capability
- use process capability

Continuously-Improving Level

When interpreting the common features for the Continuously-Improving Level, note that change is based on a quantitative understanding of the effectiveness of process changes.

Improving Organizational Capability

- establish process effectiveness goals
- continuously improve the standard process

Improving Process Effectiveness

- perform causal analysis
- eliminate defect causes
- continuously improve the defined process

Generic Practice A generic practice is an implementation or institutionalization practice that enhances the capability to perform any process.

As an example, the Tracking Performance common feature contains this generic practice (see clause 5): “2.4.1 Track with measurement. Track the status of the process against the plan using measurement.” Recording data on conditions and results while performing a process increases understanding of its performance, and addresses the "tracking performance" aspect of performing the process. This is one of many generic practices that contribute to improved capability to manage the performance of the process (the data conveys status and results to the one tracking the performance of the process).

The generic practices apply to a process as a whole and can be aggregated by common features and by capability levels to describe the capability of a process. The generic practices characterize good process management that results in an increasing process capability for any process. A planned, well-defined, measured, and continuously improving process is consistently performed as the generic practices are implemented for a process. This process capability is built on the foundation of the base practices that describe the unique, functional activities of the process.

When using the generic practices, it may be helpful to substitute the specific name of the process for the phrase “the process.” For example, when judging generic practice 2.1.1 for the "Perform configuration management" SUP.2 process, “allocate adequate resources (including people) for performing the process” becomes “allocate adequate resources (including people) for performing configuration management.”

C.3 Relationship of components

The model’s architecture, shown in figure 2, contains two hierarchies. The hierarchy on the left consists of process categories, which are composed of processes, which are composed of base practices. This is a decomposition by type of activity.

Processes are rated in terms of the right-side hierarchy. Processes may be rated at a capability level; capability levels are composed of common features (except for level 0: Not-Performed); common features, in turn, are composed of generic practices.

Combining these two hierarchies in rating a process is illustrated in figure 3. Rating processes is described in detail in part 3 of this International Standard.

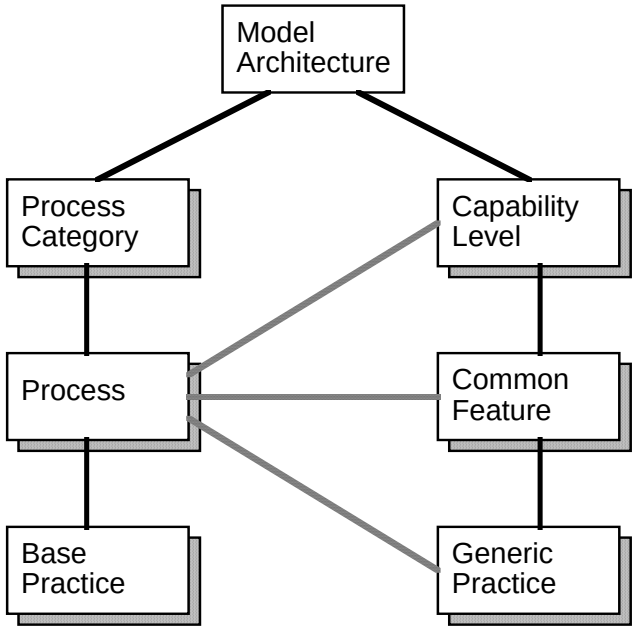

Figure 2 – Model architecture: two ways to classify practices.

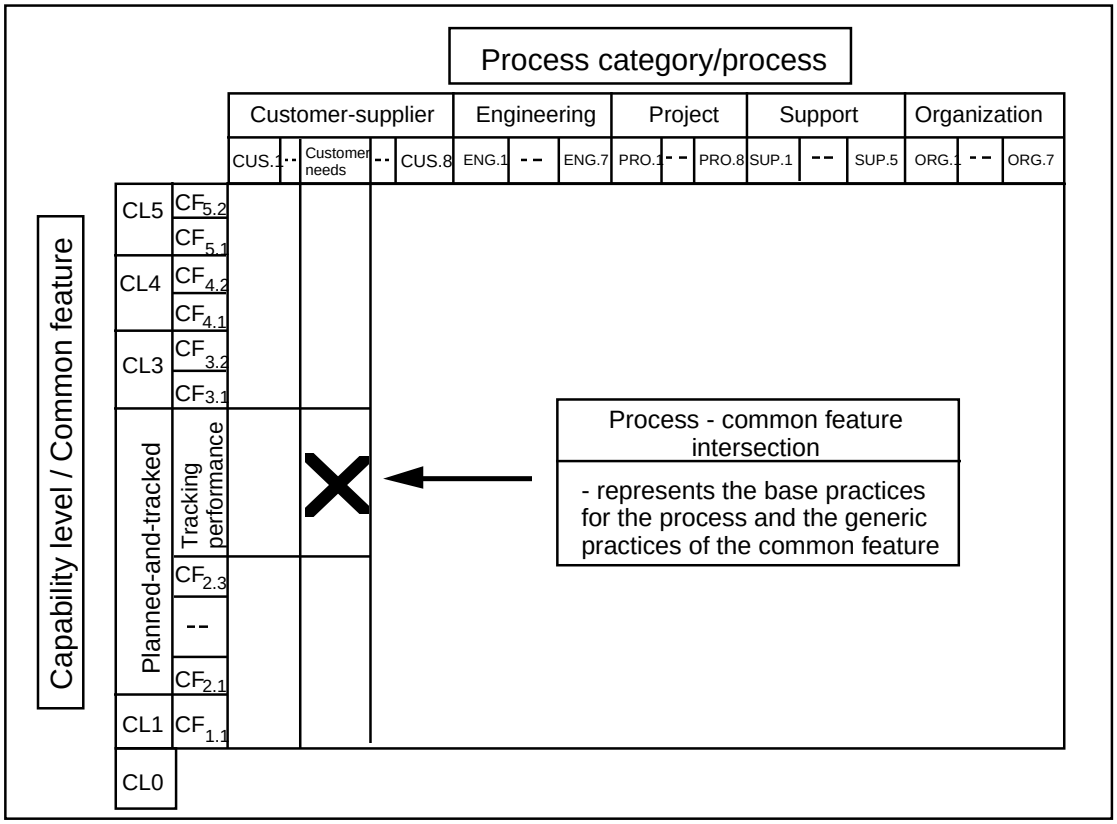

Figure 3 Architecture: 2-dimensional view

C.4 Ordering common features by capability levels

By their nature, there is more than one way to group aspects into common features and common features into capability levels. The ordering of the common features in this model stems from the observation that some implementation and institutionalization aspects benefit from the presence of others. This is especially true if institutionalization aspects are well established. For example, the provision of a well-defined, usable process for an entire organization to tailor and use should follow from some experience in managing the performance of that process at the level of individual projects. An example of this is that prior to institutionalizing a specific estimation process for an entire organization, the organization first attempts to use the estimation process on a project. Some aspects of process implementation and institutionalization should be considered together (not one ordered before the other) since they work together toward enhancing capability.

Common features and capability levels are important in performing an assessment and improving an organization's process capability. In the case of an assessment where an organization has some, but not all common features implemented at a particular capability level for a particular process, the organization usually is operating at the lowest completed capability level for that process. For example, at capability level 2, if the Tracking Performance common feature is lacking, it will be difficult to track project performance. If a common feature is in place, but not all its preceding ones (i.e.-those at lower capability levels), the organization may not reap the full benefit of having implemented that common feature. An assessment team should take this into account in assessing an organization's individual processes.

In the case of improvement, organizing the practices into capability levels provides an organization with an improvement route map should it desire to enhance its capability for a specific process. For these reasons, the generic practices are grouped into common features which are ordered by capability levels.

In either case, an assessment should determine the capability levels for all of the different processes within the assessment scope. Processes can, and probably will, exist at different levels of capability. The organization will then be able to utilize this process-specific information to focus the improvements of its processes. The priority and sequence of the improvement of the organization's software processes should take into account their business goals.

Annex D (informative)

Example process with generic practices elaborated

The following elaboration of “Establish project plan” is intended to help assessors in recognizing and understanding how to judge the adequacy of implementation of generic practices. There are many ways to implement a process, and there are interactions between processes and practices that can best be identified by discussing subtleties of implementation in a specific context. The discussion of generic practice 2.1.6, for example, identifies some of the complexities to be considered when looking at large project versus small project contexts.

PRO.2 Establish project plan

The purpose of the *Establish project plan* process is to establish reasonable plans for performing the software engineering and as a basis for managing the software project.

Project plans typically document

- project purpose and objectives;
- work products to be developed;
- software life cycle model;
- software estimates;
- project risks and mitigation plans;
- schedule;
- resources allocated to the project activities;

Inputs to this process can include the project's software life cycle model produced by process, "Plan project life cycle" PRO.1.

Project planning involves developing estimates for the work to be performed, establishing the necessary commitments, and defining the plan to perform the work. It begins with a statement of the work to be performed and other constraints and goals that define and bound the software project, those established by the practices of the “Identify customer needs” CUS.3 and “Manage requirements” PRO.4 processes. The planning process includes steps to estimate the size of the work products and the resources needed, produce a schedule, identify and assess an initial set of risks, and negotiate commitments. Iterating through these steps may be necessary to establish the plan for the project.

This plan provides the basis for performing and managing the project's activities and addresses the commitments to the software project's customer according to the resources, constraints, and capabilities of the software project.

Level 1: Performed-Informally Level

Common Feature 1.1: Performing Base Practices

1.1.1 Perform the process. Perform the base practices in implementing the process to provide work products and/or services to a customer.

Note: The customer of the process may be internal or external to the organization.

The base practices for “Establish project plan” follow.

PRO.2.1 Develop work breakdown structure. Develop a work break down structure relating project tasks and sequence with the resources required to accomplish them.

Note: This work break down structure will be critically dependent on having a statement of work for the software project that covers scope of the work, technical goals and objectives, identification of customers and end users, imposed standards, assigned responsibilities, cost and schedule constraints and goals, dependencies between the software project and other organizations, resource constraints and goals, and other constraints and goals for development and/or maintenance.

PRO.2.2 Identify project standards. Identify the software standards which will guide the project's software activities, consistent with the needs of the project.

Note: Examples of software standards and procedures include software planning, software configuration management, software quality assurance, risk management, software design, problem tracking and resolution, and software measurement.

PRO.2.3 Identify specialized facilities. Identify any specialized tools, equipment, or rooms beyond those normally available which will be needed to meet any unusual technical requirements of the project.

Note: Examples include target hardware, simulators, specialized test equipment, and test labs.

Process, "Provide software engineering environment" ORG.6 deals with providing the (normal) software engineering environment. Specialized facilities may be needed due to security requirements of the project.

Plan for the project's software engineering facilities and support tools, basing estimates of capacity requirements for these facilities and support tools on the size estimates of the software work products and other characteristics.

Examples of software engineering facilities and support tools include software development computers and peripherals, software test computers and peripherals, target computer environment software, and other support software.

PRO.2.4 Determine reuse strategy. Identify opportunities for reuse, analyze their impacts, and determine the strategy for reuse.

Note: This base practice benefits from process, "Enable reuse" ORG.5.

PRO.2.5 **Develop project estimates.** Develop estimates of what is needed to satisfy the software requirements for the entire software life cycle.

Note: Parameters to estimate include size, effort, cost, schedule, and resources

Examples of types of work products and activities for which size estimates are made include operational software and support software, deliverable and non deliverable work products, software and non software work products (e.g., documents), and activities for developing, verifying, and validating work products.

PRO.2.6 **Identify initial project risks.** Identify, assess, and document an initial (or baseline) set of project risks and their mitigation plans.

Note: Software project risks include risks to staying within budgeted software size, cost, effort, schedule; risks in availability of resources; and technical risks.

See process, "Manage risks" PRO.6.

Identify, assess, and document the software risks associated with the cost, resource, schedule, and technical aspects of the project. Analyze and prioritize risks based on their potential impact to the project. Identify contingencies for the risks. Examples of contingencies include schedule buffers, alternate staffing plans, and alternate plans for additional computing equipment.

PRO.2.7 **Identify project measures.** Identify the basic set of status and other measures which will be used to track project progress and help determine if the project is meeting its objectives.

Note: Examples of software size measurements include function points, feature points, lines of code, number of requirements, and number of pages.

PRO.2.8 **Establish project schedule.** Establish the project schedule, based on the software life cycle model, work breakdown structure, estimates, and risk mitigation plans.

Note: Examples of software life cycles include waterfall, overlapping waterfall, spiral, serial build, and single prototype/overlapping waterfall.

PRO.2.9 **Establish project commitments.** Establish commitments to the estimates and plans with all affected groups.

Note: Review software project commitments made to individuals and groups external to the organization with senior management.

PRO.2.10 Document project plans. Document the results of the activities in this process within the project plans.

Note: This includes documenting the project's software life cycle model.

See discussion of generic practice 2.1.6 later.

The software plan covers:

- The software project's purpose, scope, goals, and objectives.
- Selection of a software life cycle.
- Identification of the selected procedures, methods, and standards for developing and/or maintaining the software.
- Identification of software work products to be developed.
- Size estimates of the software work products and any changes to the software work products.
- Estimates of the software project's effort and costs.
- Estimated use of critical computer resources.
- The software project's schedules, including identification of milestones and reviews.
- Identification and assessment of the project's initial software risks.
- Plans for the project's software engineering facilities and support tools.

Level 2: Planned-and-Tracked Level

Common Feature 2.1: Planning Performance

2.1.1 Allocate resources. Allocate adequate resources (including people) for performing the process.

Note: Resource management is described in project process, "Manage resources and schedule" PRO.7.

Where feasible, make experienced individuals, who have expertise in the application domain of the software project being planned, available to develop the software plan.

Initiate software project planning in the early stages of, and in parallel with, the overall project planning.

2.1.2 Assign responsibilities. Assign responsibilities for developing the work products and/or providing the services of the process.

Note: Designate a project software manager to be responsible for negotiating commitments and developing the project's software plan.

Assign responsibilities for developing the software plan.

The project software manager, directly or by delegation, co-ordinates the project's software planning.

Partition the software work products and activities, and assign responsibilities to software managers in a traceable, accountable manner. Examples of software work products include work products for delivery to the external customer or end users, as appropriate; work products for use by other engineering groups; and major work products for internal use by the software engineering group.

The software engineering group participates on the project proposal team.

Involve the software engineering group in proposal preparation and submission, clarification discussions and submissions, and negotiations of changes to commitments that affect the software project.

Assign responsibilities and negotiate commitments to procure or develop software engineering facilities and support tools.

2.1.3 Document the process. Document the approach to performing the process in standards and/or procedures.

Note: Employee participation in developing standards and procedures is essential to creating a usable process definition. The people who perform the process are its owners.

See ORG2.9 “Establish policy” for the establishment of organizational policies that should be followed, specifically as instantiated in planning. Follow a written organizational policy for planning a software project.

Review the statement of work according to a documented procedure, which will typically specify reviews by the project manager, the project software manager, the other software managers, and other affected groups.

Develop the project's software plan according to a documented procedure, which will typically specify that:

- the software plan is based on and conforms to the customer's standards, as appropriate; the project's standards; the approved statement of work; and the allocated requirements.
- plans for software-related groups and other engineering groups involved in the activities of the software engineering group are negotiated with those groups, the support efforts are budgeted, and the agreements are documented.
- plans for involvement of the software engineering group in the activities of other software-related groups and other engineering groups are negotiated with those groups, the support efforts are budgeted, and the agreements are documented.
- the software plan is reviewed by the project manager, the project software manager, the other software managers, and other affected groups.

For PRO.2.5, derive estimates for the size of the software work products (or changes to the size of software work products) according to a documented procedure, which will typically specify that:

- size estimates are made for all major software work products and activities;
- software work products are decomposed to the granularity needed to meet the estimating objectives;
- historical data are used where available;
- size estimating assumptions are documented;
- size estimates are documented, reviewed, and agreed to. Examples of groups and individuals who review and agree to size estimates include the project manager, the project software manager, and the other software managers.

Similarly, derive estimates for the software project's effort and costs according to a documented procedure, which will typically specify that:

- estimates for the software project's effort and costs are related to the size estimates of the software work products (or the size of the changes);
- productivity data (historical and/or current) are used for the estimates when available, and sources and rationale for these data are documented.;
- the productivity and cost data are from the organization's projects when possible;
- the productivity and cost data take into account the effort and significant costs that go into making the software work products. Examples of significant costs that go into making the software work products include direct labour expenses, overhead expenses, travel expenses, and computer use costs;
- effort, staffing, and cost estimates are based on past experience;
- similar projects are used when possible;
- time phasing of activities is derived;
- distributions of the effort, staffing, and cost estimates over the software life cycle are prepared;
- estimates and the assumptions made in deriving the estimates are documented, reviewed, and agreed to.

Derive estimates for the project's critical computer resources according to a documented procedure, which will typically specify that:

- critical computer resources for the project (e.g. computer memory capacity, computer processor use, and communications channel capacity) are identified;
- estimates for the critical computer resources are related to the estimates of: the size of the software work products, the operational processing load, and the communications traffic;
- estimates of the critical computer resources are documented, reviewed, and agreed to.

Critical computer resources may be in the host environment, in the integration and testing environment, in the target environment, or in any combination of these.

For PRO.2.8, derive the project's software schedule according to a documented procedure, which will typically specify that:

- the software schedule is related to: the size estimate of the software work products (or the size of changes), and the software effort and costs;
- the software schedule is based on past experience;
- similar projects are used when possible;
- the software schedule accommodates the imposed milestone dates, critical dependency dates, and other constraints;
- the software schedule activities are of appropriate duration and the milestones are of appropriate time separation to support accuracy in progress measurement;
- assumptions made in deriving the schedule are documented;
- the software schedule is documented, reviewed, and agreed to.

2.1.4 Provide tools. Provide appropriate tools to support performance of the process.

Note: Examples of support tools for planning include spreadsheet programs, cost estimation models, and project planning and scheduling programs.

2.1.5 Ensure training. Ensure that the individuals performing the process are appropriately trained in how to perform the process.

Note: Train the software managers, software engineers, and other individuals involved in software planning **i**n the software estimating and planning procedures applicable to their areas of responsibility.

2.1.6 Plan the process. Plan the performance of the process.

Note: Project planning is described in this process, "Establish project plan" PRO.2, so this could be considered a “recursive” instantiation of planning. Generic practice 2.1.6 refers to the planning of the process, "Establish project plan" PRO.2 rather than the actual software project plan itself.

In large software projects, where millions of dollars/yen/pounds/etc., may be spent on a proposal or concept exploration phase, it is appropriate to “plan the plan.” In more common projects, this “plan the process” practice would probably be realized more at the organizational level in terms of establishing strategic directions for products and planning guidelines, which are then monitored by senior management.

Common Feature 2.2: Disciplined Performance

2.2.1 Use plans, standards, and procedures. Use documented plans, standards, and/or procedures in implementing the process.

Note: The standards and procedures used were documented in 2.1.3. The plans used were documented in 2.1.6.

2.2.2 Do configuration management. Place work products of the process under version control or configuration management, as appropriate.

Note: The process for configuration management is described in support process, "Perform configuration management" SUP.2.

The following planning work products would typically be considered

- statement of work;
- software plan;
- software planning data.

Common Feature 2.3: Verifying Performance

2.3.1 Verify process compliance. Verify compliance of the process with applicable standards and/or procedures.

Note: The applicable standards and procedures were documented in 2.1.3 and used in 2.2.1. The quality assurance process is described in support process, "Perform quality assurance" SUP.3.

At a minimum, the reviews verify conduct of the following activities according to the applicable procedures:

- activities for software estimating and planning;
- activities for reviewing and making project commitments;
- activities for preparing the software plan;

2.3.2 Audit work products. Verify compliance of work products with the applicable standards and/or requirements.

Note: Requirements are identified in Customer-Supplier process, "Identify customer needs" CUS.3, and managed in project process, "Manage requirements" PRO.4.

At a minimum, the audits verify the content of the software plan complies with the applicable standards and procedures.

Common Feature 2.4: Tracking Performance

2.4.1 Track with measurement. Track the status of the process against the plan using measurement.

Note: The use of measurement implies that the measures have been defined and selected, and data has been collected.

Note that the measures identified in PRO.2.7, “Identify project measures” are the measures used to manage the software project rather than the measures identified here for tracking the status of the planning process (PRO.2). As was discussed for 2.1.6, measurement of large project planning efforts may be implemented directly by a proposal team; more typical projects will probably aggregate and track planning overhead more indirectly.

Examples of measurements include completion of milestones for the software project planning activities compared to the plan; and work completed, effort expended, and funds expended in the software project planning activities compared to the plan.

Review the activities for software project planning with the project manager on both a periodic and event-driven basis ensuring that affected groups are represented. Topics for review include:

- status and current results of the software project planning activities against the software project's statement of work and customer requirements;
- dependencies between groups;
- conflicts and issues not resolvable at lower level;
- software project risks;
- assignment, review, and tracking of action items.

Prepare and distribute a summary report from each meeting to the affected groups and individuals.

Review the activities for software project planning with senior management on a periodic basis for the purpose of providing awareness of, and insight into, planning activities at an appropriate level of abstraction and in a timely manner. The time between reviews should meet the needs of the organization and may be lengthy, as long as adequate mechanisms for exception reporting are available. Review topics include:

- the technical, cost, staffing, and schedule performance;
- conflicts and issues not resolvable at lower levels;
- software project risks;
- assignment, review, and tracking of action items.

Prepare and distribute a summary report from each meeting to the affected groups and individuals.

2.4.2 Take corrective action. Take corrective action as appropriate when progress varies significantly from that planned.

Note: Progress may vary because estimates were inaccurate, performance was affected by external factors, or the requirements, on which the plan was based, have changed. Corrective action may involve changing the process or changing the plan or both.

Record software planning data for possible use in future planning/replanning as part of 2.4.1. Information recorded includes the estimates and the associated information needed to reconstruct the estimates and assess their reasonableness.

Level 3: Well-Defined Level

Common Feature 3.1: Defining a Standard Process (Organization-Level Common Feature)

3.1.1 Standardize the process. Document a standard process or family of processes for the organization, which describes how to implement the base practices for the process.

Note: The critical distinction between 2.1.3 and 3.1.1 is the scope of application of the standards and procedures. In 2.1.3, the standards and procedures may be used in only a specific instance of the process, e.g., in a particular project. In 3.1.1, standards and procedures are being established at an organizational level for common use. The process definition process is described in the "Organization" process category, "Define the process" ORG.2.

For “Establish project plan,” the standard process is for the planning of software projects. It may specify (or encourage) cost models, such as COCOMO, PRICE-S, and SLIM. It may specify (or encourage) structured brainstorming/consensus techniques for estimating, such as the Delphi method and nominal group technique. It may specify (or encourage) life cycle models, such as spiral or incremental development.

3.1.2 Tailor the standard process. Tailor the organization's standard process family to create a defined process which addresses the particular needs of a specific use.

Note: The phrase “which addresses the particular needs of a specific use” caters to the general case of organization-level, as opposed to project-level, processes. For defined processes at the project level, the tailoring addresses the particular needs of the project. The organization's standard process family is documented in 3.1.1 and ORG.2.

Select the models, techniques, and tools that are supported by the organization for planning software projects and which most directly address the application domain and customer needs (i.e., specific planning or reporting requirements may be imposed by the customer).

Common Feature 3.2: Performing the Defined Process

3.2.1 Use a well-defined process. Use a well-defined process in implementing the process.

Note: The defined process will typically be tailored from the organization's standard process, as described in 3.1.2. A well-defined process is one with inputs, entry criteria, tasks, validation, outputs, and exit criteria that are documented, consistent, and complete.

For "Establish project plan," this means use a well-defined planning process that results in a clear course of action with the appropriate reviews and checkpoints defined to make the planning process responsive to the changing environment.

3.2.2 Perform peer reviews. Perform peer reviews of appropriate work products of the process.

Note: The process for peer reviews is described in support process, "Perform peer reviews" SUP.5.

The software engineering group participates with other affected groups in the overall project planning throughout the project's life; specifically, they review the project-level plans.

The software engineering group reviews the project's proposed commitments. Examples of project commitments include the project's technical goals and objectives; the system and software technical solution; the software budget, schedule, and resources; and the software standards and procedures.

Estimates that are reviewed include those for size, cost, effort, critical computer resources, and schedule.

Review the risks that have been identified.

All affected groups review the plans for software engineering facilities and support tools.

3.2.3 Use well-defined data. Use data on performing the defined process to manage the defined process.

Note: This is an evolution of 2.4.2; corrective action taken here is based on a well-defined process, which has objective criteria (see 3.2.1) for determining progress.

For "Establish project plan," this implies using measurements that are defined and used across projects in the organization and using planning data (for estimating) based on historical performance and that is maintained in an organizational database.

Level 4: Quantitatively-Controlled Level

Common Feature 4.1: Establishing Measurable Quality Goals

4.1.1 Establish quality goals. Establish measurable quality goals for the work products of the organization's standard process family.

Note: These quality goals can be tied to the strategic quality goals of the organization, the particular needs and priorities of the customer, or to the tactical needs of the project.

For “Establish project plan,” this implies planning to build products that are aligned with strategic business objectives, e.g., in a particular market or product line that the organization has chosen to compete in.

Common Feature 4.2: Objectively Managing Performance

4.2.1 Determine process capability. Determine the process capability of the defined process quantitatively.

Note: This is a quantitative process capability based on a well-defined (3.1.1) and measured process.

For “Establish project plan,” this implies understanding how good the planning process is.

4.2.2 Use process capability. Take corrective action as appropriate when the process is not performing within its process capability.

Note: Special causes of variation, identified based on an understanding of process capability, are used to understand when and what kind of corrective action is appropriate. This is an evolution of 3.2.3, with the addition of quantitative process capability to the defined process.

For “Establish project plan,” this implies identifying when the planning process is not working effectively and will be realized more in replanning than in initial planning.

Level 5: Continuously-Improving Level

Common Feature 5.1: Improving Organizational Capability (Organization-Level Common Feature)

5.1.1 Establish process effectiveness goals. Establish quantitative goals for improving process effectiveness of the standard process family, based on the business goals of the organization and the current process capability.

Note: For “Establish project plan,” this implies establishing goals such as cost/performance indices that maximize the accuracy of plans within the limits of planning capability.

5.1.2 Continuously improve the standard process. Continuously improve the process by changing the organization's standard process family to increase its effectiveness.

Note: Changes to the organization's standard process family may come from innovations in technology or incremental improvements. Innovative improvements will usually be externally driven by new technologies. Incremental improvements will usually be internally driven by improvements identified during tailoring (3.1.2) or defect prevention (5.2.2) activities.

The effect of improving the standard process is to attack common causes of variation. Special causes of variation are controlled in 4.2.2. Common causes of variation across instantiations of the process are changed in this practice.

For “Establish project plan,” this deploying new estimating techniques, cost models, etc., that have been demonstrated to be effective.

Common Feature 5.2: Improving Process Effectiveness

5.2.1 Perform causal analysis. Perform causal analysis of defects.

5.2.2 Eliminate defect causes. Eliminate the causes of defects in the defined process selectively.

Note: Defect causes are selectively eliminated because it may be impractical to perform causal analysis (5.2.1) on all defects, so some screening criteria may be used. Also note the desirability of eliminating similar, as yet undiscovered, defects in the product, as well as eliminating the cause of the defect.

5.2.3 Continuously improve the defined process. Continuously improve process performance by changing the defined process to increase its effectiveness.

Note: The improvements may be based on incremental improvements (5.2.2) or new technologies (perhaps as part of pilot testing) in ORG.3.8). Improvements will typically be driven by the goals established in 5.1.1.

Annex E (informative)

Mapping to ISO 12207

E.1 Introduction

ISO 12207 is a stand-alone document which is primarily intended for use in a contract situation involving the acquisition/supply of a system that contains software. Thus for example, it contains explicit links between processes which determine a "calling sequence" of processes, and assigns responsibilities for activities/products between parties to the contract - elements which would be inappropriate for this part of the International Standard, given its differences in scope and intended use. Therefore, it is inappropriate to try to individually map every ISO 12207 process task to a particular practice in this part of the International Standard. Instead, the mapping has been performed on the ISO 12207 views, process categories and processes. Each mapping is summarized by a table, followed by a textual explanation briefly outlining the rationale for mapping each entry.

E.2 Mapping ISO 12207 views

There are five basic views in ISO 12207 - contract, management, operating, engineering and supporting. These views depict the ISO 12207 software life cycle processes and their relationships under different views of the usage of the standard. Although there is no direct equivalent in this part of the International Standard, these ISO 12207 views can be mapped as shown in table 7.

Table 7 – ISO 12207 views mapping

12207 View	Process Categories
Contract	Customer-Supplier
Management	Project
Operating	Customer-Supplier
Engineering	Engineering
Supporting	Support

E.2.1 Contract view

The contract view defines the tasks of the acquirer and supplier from the contractual viewpoint. This part of the International Standard doesn't contain an explicit supplier process, however the acquirer contract-establishing processes are in the “Customer-Supplier” process category.

E.2.2 Management view

Under the management view, the acquirer, developer, operator etc. manage their respective processes for the software project. T part of the International Standard “Project” process category consists of processes which establish the project and manage its resources. In addition, each process in this part of the International Standard has a set of common level generic practices which focus on planning and tracking the performance of the process.

E.2.3 Operating view

Under the operating view, the operator provides software operation services for the users. This part of the International Standard “Customer-Supplier” process category consists of processes that work directly with the customer, including supporting operation of the software and providing customer support.

E.2.4 Engineering view

Under the engineering view, the developer or maintainer conduct their respective tasks to produce or modify software products. This part of the International Standard “Engineering” process category consists of processes that directly specify, implement, or maintain a system and software product.

E.2.5 Supporting view

Under the supporting view, supporting services (such as configuration management and quality assurance) are provided to others in fulfilling specific, unique tasks. This part of the International Standard “Support” process category consists of process (including configuration management and quality assurance) which enable and support the performance of the other processes on a project.

E.3 Mapping ISO 12207 process categories

There are three process categories in ISO 12207 (primary, supporting and organizational) and five in this part of the International Standard (customer-supplier, engineering, project, support, and organization). The ISO 12207 process categories correspond to the process categories as shown in table 8.

E.3.1 Primary process category

The ISO 12207 Primary process category contains processes that initiate or perform development, operation and maintenance activities. It corresponds to the “Engineering” process category, although operation activities are in the “Customer-Supplier” process category since the customer is directly involved.

E.3.2 Supporting process category

The ISO 12207 Supporting process category contains processes that support other processes as an integral part with a distinct purpose and contribute to the success and quality of the software project. It corresponds to the “Support” process category, although joint reviews with the customer are in the “Customer-Supplier” process category, since again, the customer is directly involved.

E.3.3 Organizational process category

The ISO 12207 Organizational process category contains processes which are established at the organizational level to provide an infrastructure of viable processes and competent personnel. It corresponds to the “Organization” process category, although some of the management activities are in the “Project” process category, because they co-ordinate and manage the project and its resources (in addition, each this part of the International Standard process has a set of common level generic practices which focus on planning and tracking performance of the process).

Table 8 – Process category mapping

ISO 12207 Process Category	Process Category
Primary	Engineering Customer - Supplier
Supporting	Support Customer - Supplier
Organizational	Organization + Project

E.4 Mapping ISO 12207 processes

There are seventeen processes in ISO 12207 and thirty five in this part of the International Standard. ISO 12207 processes are comprised of activities and tasks, while this part of the International Standard processes consist of practices (base practices, which address the purpose of the process, and generic practices, which help to implement and institutionalize the process and enhance its capability). Some ISO 12207 activities and tasks map to base practices, while others, particularly those concerned with implementing and planning the process are covered by generic practices.

The purpose and activities of ISO 12207 processes provide a better basis than tasks for mapping to this part of the International Standard, because, given the differences in scope and intended use of the two documents, ISO 12207 tasks are generally at a lower level of detail than this part of the International Standard practices. Using this basis, the seventeen ISO 12207 processes map to this part of the International Standard as shown in table 9.

Table 9 – Process mapping

ISO 12207 Process	Processes
Acquisition	Acquire software products and/or service Establish contract
Supply	Establish contract Perform joint audits and reviews Package, deliver, and install the software Support operation of software Plan project life cycle Establish project plan Manage Project Resources Manage quality
Development	Develop software requirements Develop software design Integrate and test software Integrate and test system Package, deliver, and install the software Perform Audits and Reviews Plan project life cycle Establish project plan
Operation	Support operation of software
Maintenance	Maintain system and software
Documentation	Develop documentation
Configuration Management	Perform configuration management
Quality Assurance	Perform quality assurance
Verification	Level 2 Generic Practices Perform peer reviews
Validation	(to be completed)
Joint Review	Perform Audits and Reviews Manage Project Resources
Audit	Perform quality assurance Perform joint audits and reviews
Problem Resolution	Problem Resolution
Management	Level 2 Generic Practices Establish project plan Manage Project Resources
Infrastructure	Level 3 Generic Practices Define the process Provide Development Environment Provide work facilities
Improvement	Level 3, 4, 5 Generic Practices Define the process Improve Process
Training	Level 2 Generic Practices Training
Tailoring	Annex A Level 3 Generic Practices

E.4.1 Acquisition process

The Acquisition process contains the tasks of the acquirer, such as identifying the need to acquire a software produce/service, considering the options for acquisition, preparing a request proposal, selecting a supplier, negotiating a contract etc. The “Acquire software product and/or service” CUS.1 process covers these activities, except contract negotiation, which is covered by the “Establish contract” CUS.2 process.

E.4.2 Supply process

The Supply process contains the activities and tasks of the supplier; preparing a proposal, negotiating a contract, developing and executing plans for managing the project, co-ordinating reviews with the supplier, and supporting the delivered product. This part of the International Standard does not contain a separate Supplier process, but all of the above ISO 12207 activities are covered by processes in the Customer-Supplier and Project Process Categories.

E.4.3 Development process

The Development process contains the activities for requirements analysis, design, coding, integration, testing, installation and acceptance of the software. In this part of the International Standard, each of these activities is a process. Requirements analysis, design, coding, integration and testing processes are all part of the “Engineering” process category. Software Installation is performed through the "Package, deliver, and install the software" CUS.5 process. Support customer acceptance review is part of the "Perform joint audits and reviews" CUS.4 process. The ISO 12207 Process Implementation activity maps to the "Plan project life cycle" PRO.1 and "Establish project plan" PRO.2 processes which deal with establishing a project and managing its resources.

E.4.4 Operation process

The Operation process covers the operation of the software and operational support to users. This part of the International Standard "Support operation of software" CUS.6 process supports correct, continuing, and effective operation of the software for the duration of its intended operation.

E.4.5 Maintenance process

The Maintenance process is concerned with modifying existing software (as a result of an error, deficiency, problem or changing need) while preserving its integrity. The "Maintain system and software" ENG.7 process has a similar orientation.

E.4.6 Documentation process

The Documentation process is for recording information produced by a life cycle process or activity. The "Develop documentation" SUP.1 process deals with designing, developing, distributing and maintaining project documentation.

E.4.7 Configuration management process

The Configuration Management process applies administrative and technical procedures to control modifications and releases of identified configuration items. The "Configuration Management" process is likewise concerned with maintaining the integrity of products of the software project.

E.4.8 Quality assurance process

The Quality Assurance process is for providing adequate assurance that the processes and software products in the project life cycle conform to their specified requirements and adhere to their established plans. The "Quality Assurance" process provides appropriate visibility into the process and products of the project.

E.4.9 Verification process

The Verification process determines whether the requirements for a system or software are complete and correct. It maps to the "Peer Review" process, and is also embedded in each process through the level 2 generic practices on reviews.

E.4.10 Validation process

The Validation process determines whether the requirements and the final, as built system or software fulfils its specific intended use. This involves developing, documenting and implementing a validation plan for the project, preparing selected test requirements, test cases and test specifications for analyzing test results and testing the software as appropriate in selected areas of the target environment.

E.4.11 Joint review process

The Joint Review process is for evaluating the status and products of an activity or phase of a project and consists of project management reviews and technical reviews. In this part of the International Standard, project management and technical reviews are performed as part of the "Manage Project Resources" process. Joint reviews involving the customer are performed as part of the "Perform Audits and Reviews" process.

E.4.12 Audit process

The audit process determines compliance with the requirements, plans and contract as appropriate. In this part of the International Standard, customer audits are covered in the “Perform joint audits and reviews” CUS.4 process while other audits are covered in the “Perform quality assurance” SUP.3 process.

E.4.13 Problem resolution process

The Problem Resolution process is for analyzing and removing problems uncovered during development, operation, maintenance etc. The "Problem Resolution" process ensures that all discovered problems are analyzed and removed and trends are identified.

E.4.14 Management process

The Management process can be used by any party to manage its respective processes. In this part of the International Standard, each process has a set of level two generic practices which are concerned with planning and tracking the performance of the process. In addition to these, Project management is specifically covered by several processes in the "Project" process category, including "Establish project plan" PRO.2, which establishes the project plans and "Manage resources and schedule" PRO.7, which co-ordinates and manages the project's resources and tracks progress against the plans.

E.4.15 Infrastructure process

The Infrastructure process is used to establish and maintain the infrastructure needed for any other processes, including software, tools, techniques, standards etc. In this part of the International Standard, each process has a set of level three generic practices which deal with defining, standardizing and tailoring the process. The process "Define the process" ORG.2 is an integral part of these practices since it is concerned with building a reusable library of process definitions. The "Provide Development Environment" process deals with providing an integrated set of software development tools for use by projects. The "Provide work facilities" ORG.7 process is concerned with providing a secure and reliable environment in which software project activities may be carried out.

E.4.16 Improvement process

The Improvement process is for establishing, assessing, measuring, controlling and improving a software life cycle process. In this part of the International Standard, each process has a common set of generic practices at capability levels three, four and five, which are concerned respectively with defining, measuring/analyzing and optimizing the process. These practices are also related to the processes "Define the process" ORG.2 and "Improve the process" ORG.3.

E.4.17 Training process

The Training process is for providing and maintaining trained personnel. In this part of the International Standard, each process has a common level two generic practice which is concerned with training individuals in how to perform the process. In addition, the "Training" process is about providing the organization and projects with individuals who possess the needed skills and knowledge to perform their roles effectively.

E.4.18 Tailoring process

The Tailoring process is for performing basic tailoring of the ISO 12207 standard. Advice on tailoring this part of the International Standard is not contained within Annex A.

Annex F (informative)

Mapping to ISO 9001

This annex maps ISO 9001:1994(E) to this part of the International Standard. The mapping is done at an overview level because the ISO 9001 requirements are typically not as detailed as the practices in this part of the International Standard. The scope and the audience of the documents also differ in many essential points, such as use in contractual situations.

The mapping in table 10 covers the general requirements of ISO 9001 versus the process categories and processes. Many of the process categories, processes and practices are much more specific to software development than the ISO 9001 requirements. Because of different architectures of the documents, many of the ISO 9001 requirements are also covered by common features or generic practices.

Table 10 – ISO 9001 mapping

ISO 9001 requirements	Process categories and processes
4.1 Management responsibility	Engineer the business Manage quality (build project teams) Assess customer satisfaction
4.2 Quality system	Manage quality Perform quality assurance Define the process (Improve the process)
4.3 Contract review	Establish contract Identify customer needs Develop system requirements and design Manage risks (Perform joint audits and reviews)
4.4 Design control	Identify customer needs Establish project plan Build project teams Manage requirements Manage resources and schedule Manage risks Develop system requirements and design Develop software requirements Develop software design (Enable reuse)
4.5 Document and data control	Develop documentation Define the process
4.6 Purchasing	Manage subcontractors
4.7 Control of customer-supplied product	Develop system requirements and design (Acquire software product and/or service)
4.8 Product Identification and traceability	Develop documentation Perform configuration management Enable reuse

Table 10 (concluded)**– ISO 9001 mapping**

ISO 9001 requirements	Process categories and processes
4.9 Process control	Implement software design Provide software engineering environment Provide work facilities (Provide customer service)
4.10 Inspection and testing	Integrate and test software Integrate and test system Perform peer reviews
4.11 Control of inspection, measuring and test equipment	Integrate and test software Integrate and test system Provide software engineering environment
4.12 Inspection and test status	Integrate and test software Integrate and test system Perform configuration management (Perform problem resolution)
4.13 Control of non conforming product	Perform configuration management
4.14 Corrective and preventive action	Perform problem resolution Improve the process
4.15 Handling, storage, packaging, preservation and delivery	Package, deliver, and install the software (Support operation of software) Perform configuration management
4.16 Control of quality records	Mostly covered by common features 2.3 and 2.4 (Assess customer satisfaction) (Develop documentation)
4.17 Internal quality audits	Perform quality assurance
4.18 Training	Perform training
4.19 Servicing	Provide customer service Maintain system and software
4.20 Statistical techniques	Covered mostly by common features 4.1 and 4.2 (Improve the process)

Annex G (informative)

Derivation and traceability

The objective of this International Standard is to encourage predictable quality products, optimum productivity, and to promote a repeatable software process. To achieve this objective, the practices in this standard were derived from the many standards and software practices that have already been defined by national and international bodies such as ISO and IEEE. Relevant standards were considered to be those that address methods for improving the quality of the software process and which were recognized nationally and/or internationally by professionals in the software industry.

The following lists the sources that were used.

Bell Communications Research, Inc., *Quality Program Specification for Surveillance Management Process(SMP) - Software (General),* QPS 88.001, Issue 1.

Institute of Electrical and Electronics Engineers, Inc., *IEEE Software Engineering Standards Collection,* April 1991.

International Organization for Standardization, *Quality systems - Model for quality assurance in design, development, production, installation, and servicing*, ISO 9001 : 1994, Revised edition, July 1994.

International Organization for Standardization, *Quality management and quality assurance standards - Part 3: Guidelines for the application of ISO 9001 to the development, supply and maintenance of software,* ISO 9000-3 : 1991(E), First edition, June 1991.

International Organization for Standardization, *Quality management and quality system elements - Part 2: Guidelines for services,* ISO 9004-2 : 1991 (E), First edition, August 1991.

International Organization for Standardization, *Software Life Cycle Processes,* ISO 12207, Draft International Standard.

National Institute of Standards and Technology, 1993 *Award Criteria -Malcolm Baldrige National Quality Award,* December 1992.

M.C. Paulk, C.V. Weber, S.M. Garcia, M.B. Chrissis, and M. Bush, *Key Practices of the Capability Maturity Model, Version 1.1,* Technical Report CMU/SEI-93-TR-25, Software Engineering Institute, Pittsburgh, Pennsylvania, February 1993.

SDCE - Software Development Capability Evaluation, Dept. of the Air Force, Headquarters Air Force Materiel Command, AFMC Pamphlet 800-61, 24 Nov. 1993, Prepublication version.

SDCCR - Software Development Capability /Capacity Review, ASC Pamphlet 800-5, 11 September 1992, Department of the Air Force, Aeronautical Systems Center (AFMC).

SCE - Software Capability Evaluation Training Guide / SPA - Software Process Assessment Training Guide , Software Engineering Institute, Pittsburgh Pennsylvania.

Compita Ltd., *Software Technology Diagnostic Practitioner's Guideline,* Version 2.5, 1993.

TickIT: A Guide to Software Quality Management System Construction and Certification Using EN29001, Issue 2.0, U.K. Department of Trade and Industry and the British Computer Society, 28 February 1992.

F. Coallier, N. Gammage, A.W. Graydon, *Trillium - Software Process Self-assessment Capability Assessment* , Bell Canada, Bell Northern Research, Northern Telecom, PI Q008, Issue 4.0, March 1993.

ISO/IEC JTC1/SC7/WG7, *Information Technology Software Life Cycle Process,* Work Item 7.21, Committee Draft, March 1993.

ISO/IEC JTC1/SC7/WG8, *Information Technology: Integral Life Cycle Process,* Work Item 7.27.2, Working Draft, September 1992.

Total Quality Management, the European Model for Self-appraisal 1993, Guidelines for Identifying and addressing total quality issues, The European Foundation for Quality Management, Brussels, Belgium.

Annex H (Informative)

Style guide for extended processes

This annex describes the guidelines used in developing the process descriptions found in the base practices and processes in clause 6. These guidelines can be used by those wishing to create extended processes. See Annex A for the requirements for creating variant models.

A process description consists of several components. These guidelines provide guidance as to the format and content of each component. Nomenclature requirements are contained in clause 4.

A process description contains

- a section number;
- the name of process;
- an identifier for process (to aid in external reference);
- a first paragraph stating the purpose of the process;
- optionally, additional paragraphs;
- descriptions of its base practices.

Each base practice description contains

- an identifier for the base practice (to aid in external reference);
- the name of the base practice;
- a statement of what the base practice does;
- and optionally, a note.

For ease of use, the guidelines are contained in two tables. Table 11 contains guidelines covering the components of a process description, and table 12 covers the components of a base practice description. The guidelines applying to a particular component appear to the right of the name of that component.

Table 11 Guidelines for describing processes

Component	Guideline
Section number	1. Each process appears in its own section and is listed in the table of contents to help the reader quickly turn to the page which describes it. That is why process descriptions have a section number. When putting together a variant model containing extended processes, it may be desirable to likewise put each process description in its own section.
Name of process	1. The name of the process should identify what the process does in summary form. 2. In general, the name should be an action-oriented phrase, beginning with a verb which summarizes the action, e.g., "Plan," "Establish," "Build," etc. On the other hand if there is a widely-accepted phrase naming the process, e.g., "configuration management," in which the first word is not a verb, we recommend you add the word "Perform" to the beginning of the name. Thus you would construct the name, "Perform configuration management," in our example. 3. Name of process appears on same line as and immediately after section number.
Identifier for process	1. Processes are grouped into process categories. The identifier for a process identifies its process category and number within that category. When creating an extended process, you may wish to assign it to a process category and a number relative to the other processes in that category, for ease of reference. 2. Identifiers consist of two components: a process category abbreviation and a number. The process categories and their abbreviations are given in Annex B The abbreviations are the initial three letters of first word in the name of the category. If adding new categories, we recommend maintaining this same style for reasons of consistency, as long as no conflicting abbreviations are created. 3. Identifier for process is parenthesized and appears on same line as and immediately after name of process.
Purpose paragraph	1. States the purpose of performing the process (why perform it?). 2. Appears as the first paragraph following the section number, name, and identifier.
Additional paragraphs	1. Add additional paragraphs as appropriate that clarify what was said in the purpose paragraph, describe inputs/outputs to other processes, describe when the process is invoked, and/or give terminology.

Table 12 – Guidelines for describing base practices

Component	Guideline
Identifier	1. To ease external reference, each base practice is given an identifier. The identifier should consist of two components, the first giving the identifier of the process the base practice belongs to, and the second component giving its textual position in the description of the process. For example the first base practice appearing in the first process description of Section 6 is given the identifier, "CUS.1.1," because "CUS.1" identifies the process and it is the first base practice.
Name	1. The name of the base practice should identify what the process does in summary form. 2. The name should be an action-oriented phrase, beginning with a verb which summarizes the action, e.g., "Evaluate," "Select," "Identify," etc. The verb should be followed by a noun (or noun phrase) which indicates the object of that action. 3. Name of base practice appears on same line as and immediately after its identifier.
Statement of what it does	1. The key part of a base practice description is the statement of what it does. It should be a complete sentence. 2. The statement begins immediately after and on the same line as the identifier and name.
Note	1. In one or more paragraphs following the statement, additional information may be provided on the base practice to help in understanding what the base practice is saying.