INTERNATIONAL STANDARD
ISO/IEC 12207
SOFTWARE LIFE CYCLE PROCESSES

Raghu Singh
Federal Aviation Administration
Washington, DC, USA

BACKGROUND

In 1987 the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) established a Joint Technical Committee (JTC1) on Information Technology. The scope of the JTC1 is "Standardization in the field of information technology systems (including microprocessor systems) and equipments."

In June 1989, the JTC1 initiated the development of an International Standard, ISO/IEC 12207 [1], on software life cycle processes to fill a critical need. Since the "cottage" industry era of the late 1970's, software has been establishing itself as an integral part of many scientific and business disciplines. However, environments for developing and managing software have been proliferating for lack of a uniform framework for managing and engineering software. The International Standard fills that critical need by establishing a common framework that can be used by software practitioners to manage and engineer software. Besides, such a uniform framework would promote international trade in software products and services.

The International Standard was published August 1, 1995. The following countries participated in the development of the standard: Australia, Brazil, Canada, Czech Republic, Denmark, Finland, France, Germany, Ireland, Italy, Japan, Korea, The Netherlands, Spain, Sweden, the United Kingdom, and the United States of America.

The standard is voluntary; that is, it does not in itself impose any obligation upon anyone to follow it. Yet, it may be imposed by an organization through internal policy directive or by individual parties through contractual agreements. The standard is designed for use by one or more parties as the basis of an agreement or in a self-imposed way.

BASIC CONCEPTS

Software life cycle architecture. The standard establishes a top-level architecture of the life cycle of software. The life cycle begins with an idea or a need that can be satisfied wholly or partly by software and ends with the retirement of the software. The architecture is built with a set of processes and interrelationships among these processes. The derivation of the processes is based upon two basic principles: modularity and responsibility.

Modularity. The processes are modular; that is, they are maximally cohesive and minimally coupled to the practical extent feasible. An individual process is dedicated to a unique function.

Responsibility. A process is considered to be the responsibility of a party in the software life cycle. In other words, each party has certain responsibilities. Responsibility is one of the key principles of total quality management, as discussed later.) This is in contrast to a "text book approach," where the life

cycle functions could be studied as topics or subjects, such as management, design, measurement, quality control, etc.

The life cycle processes. The processes are grouped into three broad classes: primary; supporting; and organizational. See Figure 1 [2]. Primary processes are the prime movers in the life cycle; they are acquisition, supply, development, operation, and maintenance. Supporting processes are documentation, configuration management, quality assurance, joint review, audit, verification, validation, and problem resolution. A supporting process supports another process in performing a specialized function. Organizational processes are management, infrastructure, improvement, and training. An organization may employ an organizational process to establish, control, and improve a life cycle process.

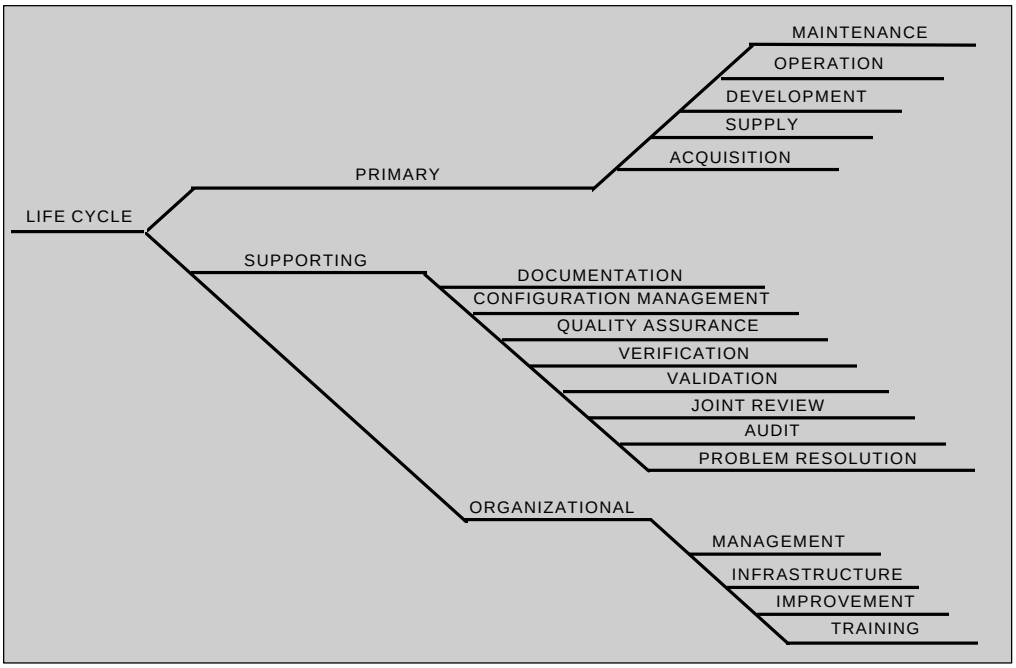

Figure 1. The Life Cycle Processes

The structure of a life cycle process. Each process is further designed in terms of its own constituent activities, each of which is further designed in terms of its constituent tasks. See Figure 2 [2]. An activity under a process is a set of cohesive tasks.

Nature of tasks. A task is a set of elementary or atomic actions. A task consumes inputs (data, information, control) and produces outputs (data, information, control). It is expressed in the form of self-declaration, requirement, recommendation, or permissible action. For this purpose, the standard carefully employs certain auxiliary verbs (will, shall, should, and may) to differentiate between the distinct forms of a task. "Will" is used to express a self-declaration of purpose or intent by one party, "shall" to express a binding provision between two or more parties, "should" to express a recommendation among other possibilities, and "may" to indicate a course of action permissible within the limits of the standard. "Must," which denotes a mandatory action, is never used in the standard.

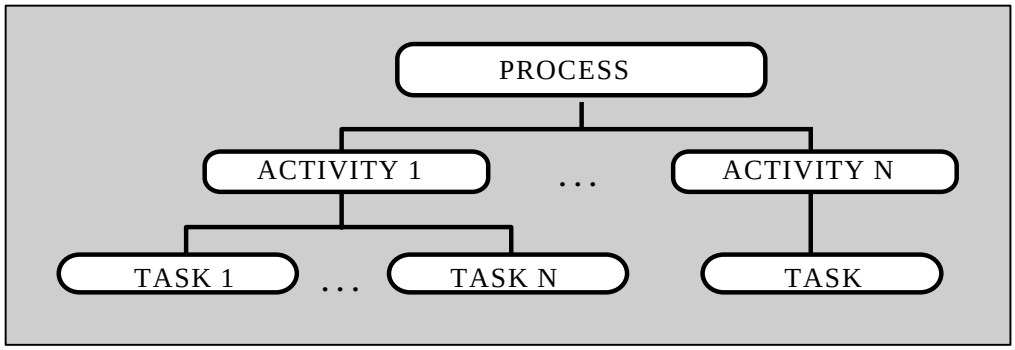

Figure 2. The Structure of a Process

There are other actions that do not contain a "shall" or a "will." These actions are not requirements; they are a preamble, an assumption, or statements that supplement or complement the context.

Nature of evaluations. In the standard, evaluation is an elementary function and used in several manners by the processes. Evaluations are conducted on various entities with given purposes against given criteria. An entity may be a process, an activity, or a task; a plan, an agreement, or a report; data, information, or products; or other. A purpose may be checking, reviewing, auditing, verifying, validating, assuring, or improving, where the motive and objective behind the respective underlying evaluations vary in practice. A criterion may be, for example, traceability of design to its requirements or correctness of design.

Total quality management. The standard implements the total quality management principles.

To begin with, the standard treats all activities, including those related to quality, as an integral part of the software life cycle. Thus, quality is automatically considered from the outset.

As the next step, the quality related activities in the life cycle are appropriated to each process. Each process is equipped with a built-in "plan-do-check-act" (PDCA) cycle. Thus, each process and the personnel responsible for performing the process are assigned their germane process-internal quality related activities, including evaluations.

In addition, two special processes, verification and validation, are provided to supplement or complement the aforementioned process-internal evaluations with desired degree of independence and objectivity.

A particular process, quality assurance, is dedicated to assuring conformity of products and services with their specified requirements. Persons responsible for this process are provided with organizational freedom and authority to effect the conformity. Organizational freedom means the freedom from the one who has the direct management responsibility for producing the product or providing the service; while authority means the authority to permit related evaluations and initiate related corrective actions. This process is not granted the authority to "force" those corrective actions.

Finally, the standard provides an improvement process for managing, controlling, and improving the established processes and their performance -- beyond contractual obligations.

Link between system and software. The standard establishes a strong link between system and software.

First, the standard is based upon the principles of general system engineering. The basic components of system engineering (such as analysis, design, fabrication, evaluation, testing, integration, quality assurance/control, manufacturing, storage/distribution, etc.) form the foundation for software engineering in the standard. For software engineering, "fabrication" is interpreted as "coding."

Second, the standard provides the minimum system context for software. Software is treated as an integral part of the total system and performs certain functions in that system. This is implemented by extracting the software from the system, producing it (concurrently with other system components), and integrating it back into the system.

Organization and party. In the standard, the terms "organization" and "party" are nearly synonymous. An organization is a body of persons organized for some specific purpose, such as a club, union, corporation, or society. When an organization, as a whole or a part, enters into an agreement, it is a party. Organizations are separate bodies, but the parties may be from the same organization or from separate organizations.

An organization or a party derives its name from the process for which it is responsible; for example, it is called an acquirer when it performs the acquisition process. A party's name is in the functional sense and does not imply a structure for an organization.

Applicability to organizations. The processes in the standard form a comprehensive set to serve various organizations. An organization, small or large, depending on its business purpose, can select an appropriate subset of the processes (and associated activities and tasks) to fulfill that purpose. An organization may perform one process or more than one process. A process may be performed by one organization or more than one organization.

The standard is intended to be applied by an organization internally or contractually by two or more organizations. In order to facilitate application of the standard either internally or contractually, the tasks are expressed in contractual language. When applied internally, the contractual language is interpreted as a self-imposed task.

Applicability to projects. The standard is written for a general, large and complex software project. The standard, however, is designed so as to be tailorable or adaptable for a software project of lesser size or complexity. It is also designed to be used whether the software is a stand-alone entity, or an embedded or integral part of a parent system.

On the same project, the standard may be applied more than once. For example, in a given software development project: an acquirer tasks a supplier to perform software development; and the supplier tasks its sub-contractor to perform all or parts of the software development. In the former, the acquirer and the supplier execute one application of the standard. In the latter, the supplier (as an acquirer) and its sub-contractor (as a supplier) execute a separate application of the standard.

Responsiveness to evolving technologies. The standard is responsive to the rapidly evolving software engineering discipline. It accomplishes this by providing a top-level, open architecture of the life cycle of software.

From a top-level viewpoint, the activities and tasks of a software life cycle process are "what-to-do" items, not "how-to-do" items. In other words, a task might be "develop an architectural design," but not "develop architectural design by using the top-down, functional-design method." This scheme provides an acquirer

an avenue to specify an end product or service and, at the same time, allows the supplier to be creative and to employ appropriate methods, techniques, and tools to produce the product or provide the service.

The standard is usable as a software development methodology or a prototyping methodology. It can be used to develop software that would serve as an application system or a prototype system. The standard can be used also as a methodology in itself to aid in analytical, feasibility, modeling, simulation, or prototyping studies of requirements, design, testing, operation, or maintenance of a system, whether or not the system would contain software. For example, operational scenarios may be prototyped and executed in a computer to gain more insight than would otherwise be not possible.

The standard is flexible and usable with: any life cycle model (such as, Waterfall, incremental, evolutionary, Spiral, or other); any software engineering method (object-oriented design, structured coding, top-down testing, or other); or any programming language (Ada, assembly, machine, or other). These are very much dependent upon the software project and state-of-the-technology, and their selection is left to the user of the standard.

The standard is adaptable by a business sector (military, commercial, automobile, airline, or other) or any national or organizational culture.

Non-prescription of events and milestones. In the standard, the processes and their activities and tasks are arranged in their most general, natural positional sequence. This positional sequence does not prescribe or dictate any time-dependent sequence. For lack of consensus on or universality of a particular time-dependent sequence, the user of the standard is left with the option to select, tailor, and order the processes, activities, and tasks as appropriate and effective. The standard encourages iteration among the activities and recursion within an activity to offset the effects of any implied sequence of activities and tasks.

Documentation of outputs. The standard requires certain outputs be documented. It does not, however, specify format, content, or media of documents. An organization may use its existing documentation sets or standards with ISO/IEC 12207. This can be implemented by establishing correlation between the calls for documentation in this standard and the organization's documentation standards.

Baselining. The standard requires that baselines of software requirements, software design, and software code be established at appropriate time(s) as planned. Baselining, when used judiciously, is an effective means for establishing confidence in milestones and controlling costs and schedules by inhibiting unnecessary, unplanned, or open changes to requirements, design, and code. The standard asks for baselining during a joint review or an audit in order to expedite and solidify acquirer-supplier understanding. A project, however, may elect not to perform baselining. The responsibility for baselining is assigned to the Development Process (and the Maintenance Process), not to the Configuration Management Process.

Software metrics. The standard is not a software metrics standard. The standard requires specification of management indicators (such as cost expenditure) and software attributes (for example, reliability, maintainability, etc.), but it does not define or prescribe them. The standard references ISO/IEC 9126 [3] for guidance on software quality characteristics. Such specification details are left to the users of the standard.

Compliance with the standard. The standard provides for compliance at the project and organizational levels, but is silent on absolute level.

At the project level, a software project tailors appropriate processes, activities, and tasks, which are performed in accordance with contractually established criteria. Once the project (including its plans and life cycle models) is established, the standard itself stays on the sidelines.

At the organizational level, an organization declares public, as a condition of trade, a set of processes, activities, and tasks from the standard, with which suppliers to the organization comply.

Once the standard is applied on a project, whether within an organization or between organizations, certain clauses are mandatory. Clauses 1, 2, and 3 are not subject to tailoring; only clauses 5, 6, and 7 are subject to tailoring. Clause 1.4, paragraph 1, invokes the Tailoring Process. Therefore, the Tailoring Process is not subject to tailoring, and the 5 “shalls” in this process are applicable to all projects.

At the absolute level, the standard is silent. Absolute compliance means every task with a "shall" or a "will" is performed. The rationale for silence on absolute compliance is that it may not be realistic for an organization to possess all the life cycle processes of the standard.

In accordance with clause 1.4 of the standard, a country may elect to establish its own compliance rules with respect to the standard. The USA has established these compliance rules in Annex F.

Certification to the standard. The standard does not address certification of an organization to the life cycle processes, nor does it define any certification criteria. It may not be realistic for an organization to possess all the processes of the standard.

Limitations. The standard is not a substitute for systematic, disciplined management and engineering of software systems. The standard merely provides a framework where the processes, activities, and tasks related to software can be reasonably identified, planned, and acted upon.

One key point to remember is that the standard contains only a set of well-defined building blocks (processes); the user of the standard should select, tailor, and assemble these processes and their activities and tasks as appropriate and cost-effective for the organization and the project.

Prerequisites to using the standard. The standard covers the full life cycle of a software system and caters to diverse, independent parties in the life cycle. The standard accommodates and reconciles the different, sometimes conflicting, objectives of the life cycle phases and of the parties. Consequently, it is inherently complex. Reading the standard outside the context of an organization's objectives and a project's needs may negatively impact proper interpretation of the standard. For effective and productive use of the standard, the following prerequisites (in that order) should be met:

a. Trained personnel;
b. Familiarity with the organization's policies;
c. Familiarity with the project's environment;
d. An understanding of the standard.

ORGANIZATION OF THE STANDARD

The International Standard begins with an Introduction that contains the reason for and the basic intent of the standard. The technical material is organized into 7 sections and 4 annexes:

Section 1 - Scope and field of application;

Section 2 - Normative references;
Section 3 - Definitions;
Section 4 - Top-level overview of the life cycle processes;
Section 5 - Activities and tasks of the five primary processes;
Section 6 - Activities and tasks of the eight supporting processes;
Section 7 - Activities and tasks of the four organizational processes;
Annex A - Activities and tasks for tailoring (or adapting) the standard for a software project;
Annex B - Brief guidance on tailoring the standard;
Annex C - General information on the processes, organizations, and their relationships;
Annex D - Bibliography.

THE PRIMARY PROCESSES

The International Standard describes a set of primary processes that occur during one period or another in the life cycle of a software project extending from its conceptualization through its retirement. The primary processes serve the key parties involved in the acquisition, supply, development, operation and maintenance of software. Each primary process is defined and described in terms of its constituent activities and tasks.

Each primary process begins with a preamble (not a requirement), follows on with a set of corporate-level actions (not requirements), and continues with a set of activities and associated tasks for providing software products and services.

Acquisition Process. This life cycle process defines the activities and tasks of the acquirer, that contractually acquires software product or service. The organization having the need for a product or service may be the owner. The owner may contract all or parts of the acquisition tasks to an agent. The acquirer represents the needs and requirements of the users.

The acquisition process begins with the definition of the need to acquire a software product or service. The process continues with the preparation and issuance of a request for proposal, selection of a supplier, and management of the acquisition process through the acceptance of the system.

This process consists of the following activities along with their specific tasks: Initiation; Request-for-Proposal preparation; Contract preparation and update; Supplier monitoring; and Acceptance and completion. The first three activities occur prior to the agreement, the last two after the agreement.

Supply Process. This life cycle process contains the activities and tasks of the supplier. The process may be initiated either by a decision to prepare a proposal to answer an acquirer's request for proposal or by signing and entering into a contract or an agreement with the acquirer to provide a software service. The service may be the development of a software product or a system containing software, the operation of a system with software, or the maintenance of a software product. The process continues with the identification of procedures and resources needed to manage and assure the service, including development and execution of plans through delivery of the service to the acquirer.

The supply process consists of the following activities along with their specific tasks: Initiation; Preparation of response; Contract; Planning; Execution and control; Review and evaluation; and Delivery and completion. The first two activities occur prior to the agreement, the last five after the agreement.

Development Process. This life cycle process contains the activities and tasks of the developer of software. The term development denotes both development of new software and modification to an existing

software. The development process is intended to be employed in at least two ways: (1) As a methodology for developing prototypes or for studying the requirements and design of a product or (2) As a process to produce products. This process provides for developing software as a stand-alone entity or as an integral part of a larger, total system.

The development process consists of the following activities along with their specific tasks: Process implementation; System requirements analysis; System design; Software requirements analysis; Software architectural design; Software detailed design; Software coding and testing; Software integration; Software qualification testing; System integration; System qualification testing; Software installation; and Software acceptance support.

The positional sequence of these activities does not necessarily imply a time order. These activities may be iterated and overlapped, or an activity may be recursed to offset any implied or default Waterfall sequence. All the tasks in an activity need not be completed in the first or any given iteration, but these tasks should have been completed as the final iteration comes to an end. These activities and tasks may be used to construct one or more developmental models (such as, the Waterfall, incremental, evolutionary, the Spiral, or other, or a combination of these) for a project or an organization.

An example of the organization of a system is depicted in Figure 3 [2]. The figure shows the system organized, at the first level, into a hardware, a software, and a set of manual operations items. At the second level each item is depicted with its components, which are further organized as needed. The organization may be achieved by partitioning the system along the top-down paths as shown. However, the bottom-up integration may not, in all cases, follow the same paths.

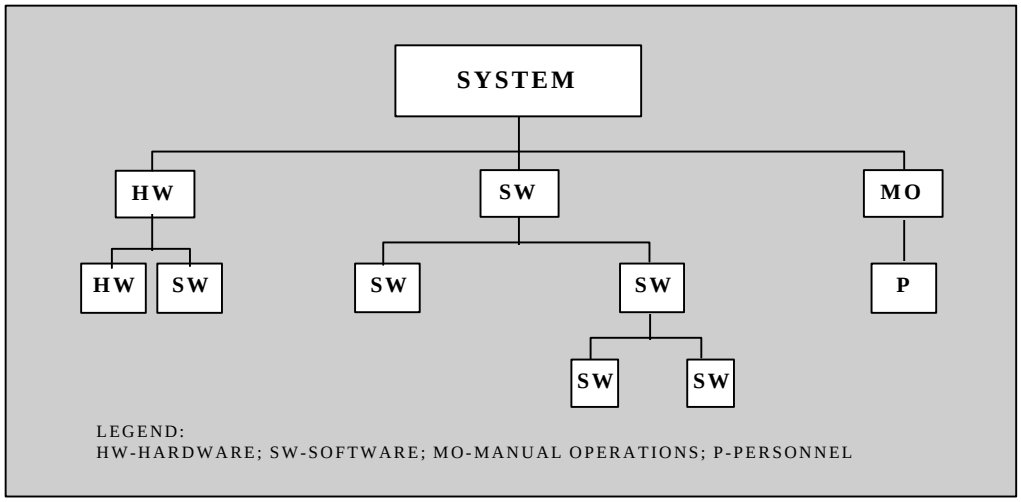

Figure 3. An example of the Organization of a System

The standard allows for baselining requirements, design and code at pre-determined points during the development of the product -- but within the control of the development process. Timely baselining inhibits premature or unplanned changes to these requirements and promotes effective change control. The standard also provides the forums (that is, the joint review and audit processes) for the interested parties to be involved in baselining.

It should be noted that the development process governs the configuration management process and baselining tasks.

Operation Process. This life cycle process contains the activities and tasks of the operator of a software system. The operation of the software is integrated into the operation of the total system. The process covers the operation of the software and operational support to users.

This process consists of the following activities along with their specific tasks: Process implementation; Operational testing; System operation; and User support.

Maintenance Process. The maintenance process contains the activities and tasks of the maintainer. This process is activated when a system undergoes modifications to code and associated documentation due to an error, a deficiency, a problem, or the need for an improvement or adaptation. The objective is to modify an existing system while preserving its integrity. Whenever a software product needs modifications, the development process is invoked to effect and complete the modifications properly. The process ends with the retirement of the system.

This process consists of the following activities along with their specific tasks: Process implementation; Problem and modification analysis; Modification implementation; Maintenance review/acceptance; migration; and Software retirement.

THE SUPPORTING PROCESSES

This standard contains a set of eight supporting processes. A supporting process supports any other process as an integral part with a distinct purpose and contributes to the success and quality of the project. A supporting process is invoked, as needed, by the acquisition, supply, development, operation or maintenance process, or another supporting process.

The supporting processes begin with a preamble, may follow on with a set of corporate-level actions (not requirements), and continue with a set of activities and associated tasks that support other life cycle processes.

Documentation Process. This is a process for recording information produced by a life cycle process. The process defines the activities, which plan, design, develop, edit, distribute and maintain those documents needed by all concerned such as managers, engineers and users of the system. The four activities along with their tasks are: Process implementation; Design and development; Production; and Maintenance.

Configuration Management Process. This process is employed to identify, define, and baseline software items in a system; to control modifications and releases of the items; to record and report the status of the items and modification requests; to ensure the completeness and correctness of the items; and to control storage, handling and delivery of the items.

This process consists of: Process implementation; Configuration identification; Configuration control; Configuration status accounting; Configuration evaluation; and Release management and delivery.

Quality Assurance Process. This process provides the framework for independently and objectively assuring (the acquirer or the customer) of compliance of products or services with their contractual requirements and adherence to their established plans. To be unbiased, software quality assurance is

provided with the organizational freedom from persons directly responsible for developing the products or providing the services.

This process consists of: Process implementation; Product assurance; Process assurance; and Assurance of quality systems.

Verification Process. This process provides the evaluations related to verification of a product or service of a given activity. Verification determines whether the requirements for a system are complete and correct and that the outputs of an activity fulfill the requirements or conditions imposed on them in the previous activities. The process covers verification of process, requirements, design, code, integration, and documentation. Verification does not alleviate the evaluations assigned to a process; on the contrary, it supplements them.

Validation Process. Validation determines whether the final, as-built system fulfills its specific intended use. The extent of validation depends upon the project's criticality. Validation does not replace other evaluations, but supplements them.

Verification or validation may be conducted by the acquirer, the supplier, or an independent party. When they are executed by an organization independent of the supplier or developer, they are called independent verification and validation (IV&V) Process.

Joint Review Process. This process provides the framework for interactions between the reviewer and the reviewee. They may as well be the acquirer and the supplier respectively. At a joint review, the reviewee presents the status and products of a life cycle activity of a project to the reviewer for comment (or approval). The reviews are at both management and technical levels.

Audit Process. This process provides the framework for formal, contractually established audits of a supplier's products or services. At an audit, the auditor assesses the auditee's products and activities with emphasis on compliance to requirements and plans. An audit may well be conducted by the acquirer on the supplier.

Problem Resolution Process. This process provides the mechanism for instituting a closed-loop process for resolving problems and taking corrective actions to remove problems as they are detected. In addition, the process requires identification and analysis of causes and reversal of trends in the reported problems. The term "problem" includes non-conformance.

THE ORGANIZATIONAL PROCESSES

This standard contains a set of four organizational processes. An organization employs an organizational process to perform functions at the organizational, corporate level, typically beyond or across projects. An organizational process may support any other process as well. These processes help in establishing, controlling, and improving other processes.

Management Process. This process defines the generic activities and tasks of the manager of a software life cycle process, such as the acquisition process, supply process, operation process, maintenance process, or supporting process. The activities cover: Initiation and scope definition; Planning; Execution and control; Review and evaluation; and Closure.

Even though, the primary processes, in general, have similar management activities, they are sufficiently different at the detailed level because of their different goals, objectives, and methods of operations. Therefore, each primary is an instantiation (a specific implementation) of the management process.

Infrastructure Process. This process defines the activities needed for establishing and maintaining an underlying infrastructure for a life cycle process. This process has the following activities: process implementation; Establishment of the infrastructure; and Maintenance of the infrastructure. The infrastructure may include hardware, software, standards, tools, techniques, and facilities.

Improvement Process. The standard provides the basic, top-level activities that an organization (that is, acquisition, supply, development, operation, maintenance, or a supporting process) needs to assess, measure, control, and improve its life cycle process. The activities cover: Process establishment; Process assessment; and Process improvement. The organization establishes these activities at the organizational level. Experiences from application of the life cycle processes on projects are used to improve the processes. The objectives are to improve the processes organization-wide for the benefit of the organization as a whole and the current and future projects and for advancing software technologies.

Training Process. This process may be used for identifying and making timely provision for acquiring or developing personnel resources and skills at the management and technical levels. The process requires that a training plan be developed, training material be generated, and training be provided to the personnel in a timely manner.

TAILORING PROCESS

Annex A, which is normative, contains the activities needed to perform a first-level tailoring for projects. Tailoring in the standard is deletion of non-applicable or in-effective processes, activities, and tasks. A process, an activity, or a task, that is not contained in the standard but is pertinent to a project, may be included in the agreement or contract. The standard requires that all the parties that will be affected by the application of the standard be included in the tailoring decisions. It should be noted that this process itself, however, cannot be tailored.

GUIDANCE

Annex B, which is informative, contains general guidance on tailoring the standard for projects. It lists the key factors upon which tailoring decisions may be made. Some of those factors are: acquisition strategy, project life cycle models, and system and software characteristics.

Annex C, which is informative, clarifies relationships among the processes and organizations.

INTERACTIONS AMONG THE PROCESSES

Figure 4 [2] depicts all the processes and their interactions. The figure is self explanatory with the following clarification.

The top box contains the organizational processes. The middle box shows the processes as they are generally applied in a project. The bottom box shows three supporting processes and the tailoring process. Clockwise arrows on an ellipse indicate plan-do-check-act (PDCA) cycle.

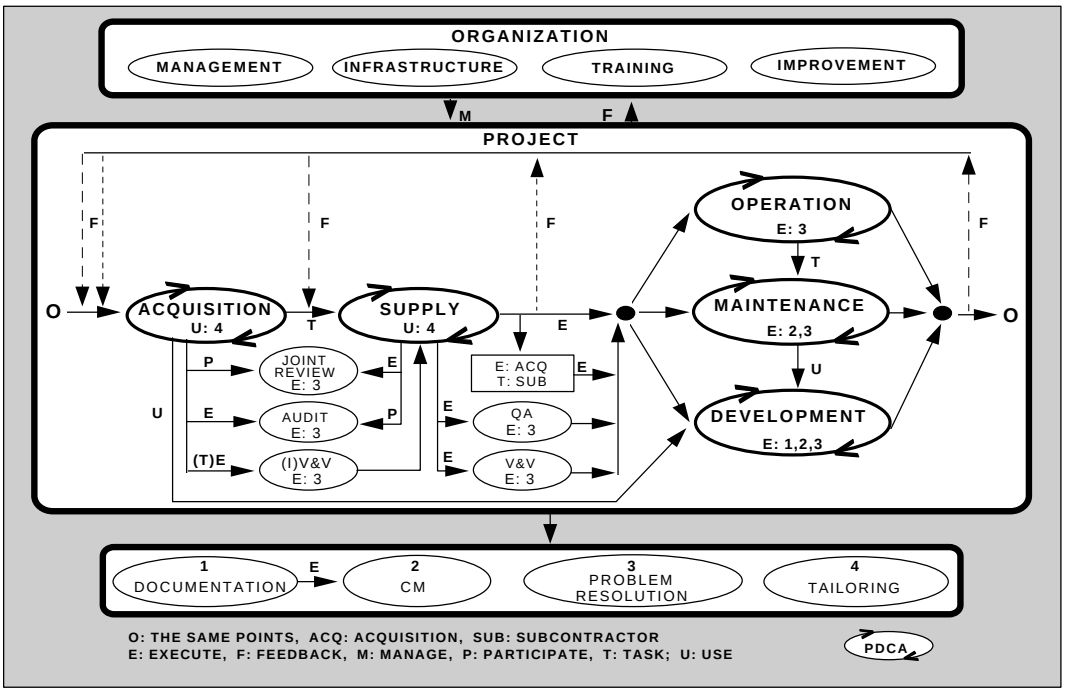

Figure 4. The Processes and their Interactions

In the middle box, the "Os" at the beginning and at the end are the same point. Letters "E, F, M, P, T, and U" mean “execute, feedback, manage, participate, task, and use” (all verbs) respectively. "(I)V&V" stands for (independent) verification and validation processes, "QA" for quality assurance process, "ACQ" for acquisition process, and "SUB" for sub-contractor. Symbol "E.n" means "execute the process numbered ‘n’ as depicted in the bottom box.”

The arrows to the right of "SUPPLY" and the little box below and right of "SUPPLY" are explained as follows. The supply process has the option to execute (E) the operation, development, or maintenance process itself. As another option (see the little box), the supplier executes the acquisition process (E: ACQ) and tasks the sub-contractor (T: SUB) to execute (E) the operation, maintenance, or development process. The arrow (U) from ACQUISITION to DEVELOPMENT indicates the acquisition process using the development process to determine system or software requirements.

APPLICATION ON A PROJECT

This section contains the factors that are important to applying ISO/IEC 12207 on projects. There are no normative rules for applying and tailoring the standard; therefore, no guidance is provided as to which clauses of the standard should be kept, deleted, or modified under these factors. At the best, such guidance may be an opinion or based on limited experience. Some helpful guidance, however, may be found in the ISO/IEC 12207 Guidebook or other literature.

Factors in the application of ISO/IEC 12207. The following factors should be considered when applying the standard. These factors are not exhaustive, neither are they listed in any particular order. It is advisable to iterate among these factors to offset any implied order. In this section, the term "user" means the user (that is, the person, party, or organization) of ISO/IEC 12207.

a. Role in the life cycle
b. Organizations' policies
c. System life cycle
d. Developmental models
e. Types of software
f. Documentation
g. Evaluations
h. Project characteristics.

Role in the life cycle. The user of the standard should determine whether it is an acquirer, a supplier, a developer, an operator, or a maintainer. This helps in determining its place, role, responsibilities, and interactions with other parties, if any, in the life cycle.

Organizations' policies. The policies of the organizations involved should be identified and analyzed for relevancy to the project. It is important to identify and analyze also those National laws and regulations on public safety, health and environment that are applicable in any case.

System life cycle. In general, a system life cycle covers the phases of: needs determination and demonstration; development; production; use; and disposal or retirement. Knowledge of the current and upcoming phases is important to the determination of the applicable primary process(es) of ISO/IEC 12207, such as the acquisition, supply, development, operation, or maintenance process.

Developmental model(s). One or more developmental models appropriate for the project should be determined. Examples of models are the Waterfall, incremental, evolutionary, reengineering, and the Spiral model. A project may need a combination of these models, or different models for different phases, builds, or portions of the product or service. The selected model(s) should be constructed with the processes and activities in the standard. The processes and activities may be recursed, revisited, overlapped, or iterated. It should be noted that, because of the presence of system-level activities, the developmental model(s) need to be compatible and consistent with the project's or the system's life cycle model.

Types of software. Types of software applicable in the project should be determined. Examples of software types are: new software, reuse of existing software "as is" or with modification, firmware, embedded software, or stand-alone software, or a combination of these. It should be noted that these types need different decisions and treatments in the life cycle.

Documentation. ISO/IEC 12207 provides for a range of outputs from its activities and tasks. It should be determined which outputs are needed, how they will be combined, packaged, and distributed, and which ones are intermediate or final products. It should be ensured that the operation and support personnel (current or future) are involved in determining the documentation needs. It should be noted that existing documentation standards of an organization are usable within the context of ISO/IEC 12207.

Evaluations. ISO/IEC 12207 has several processes, activities, and tasks that are based on (elementary) evaluations. Evaluations are primarily conducted within a process or between the processes; they are process-internal evaluations and quality assurance, verification, validation, joint review, audit, and improvement processes. It should be determined which ones are needed along with related schedules and responsibilities.

Project characteristics. Above all, the requirements and specifications of the product or service dominate the determination and selection of the processes, activities, and tasks. In addition, other characteristics are important; for example, product size, criticality, and complexity; personnel size and diversity; operation and support scenarios; public visibility; and the expected life of the product or service.

Clauses related to engineering. The user of the standard should exercise caution and technical judgment when deleting an engineering related clause. This includes the documentation that will be needed by the engineers, who will operate and maintain the products in the future.

Clauses related to management. Most of the management related clauses are covered by the execution and control activities and the quality assurance, verification, validation, joint reviews, and audit processes. The degree of independence, including organizational freedom, depends upon the degree of objectivity needed. Large, critical, or visible projects need significant management attention. For small or non-critical projects, significant management activities may be costly in time and money.

Clauses related to documentation. The extent and scope of documentation depends primarily upon the size, diversity and needs of the users and the life of the products or services. Shaving documentation during the development process may save cost and time immediately, but may be expensive for the operation and maintenance processes by orders of magnitude. Operation and maintenance personnel should be involved in determining the extent and scope of documentation.

APPLICATION IN AN ORGANIZATION

This section contains general guidance on applying ISO/IEC 12207 in organizations. An organization may be one person or several persons. Some helpful guidance, however, may be found in an upcoming ISO/IEC 12207 Guidebook.

Each primary and supporting process in the standard begins with a preamble. The preamble contains actions (in present tense) that are intended to be performed by organizations internally and beyond agreements. These actions in the preamble reference or invoke the organizational processes for details.

An organization should manage its processes in accordance with the management process, establish the infrastructure under the processes according to the infrastructure process, provide training to its personnel per the training process, and improve the processes following the improvement process.

Of particular importance is the improvement process. Once the organization established the infrastructure of the needed processes and of personnel training, it performs continuous improvement based on the applications of these processes. Figure 5 [2] shows the steps for continuous process improvement at the organizational level. The figure depicts the three activities of the improvement process: process establishment; process assessment; and process improvement.

RELATED PRODUCTS AND ACTIVITIES

Guidebook on ISO/IEC 12207. The JTC1 is in the process of developing a short guidebook on ISO/IEC 12207. This guidebook will provide, among other topics, the basic concepts behind the standard and application and tailoring guidance for use at both project and organizational levels.

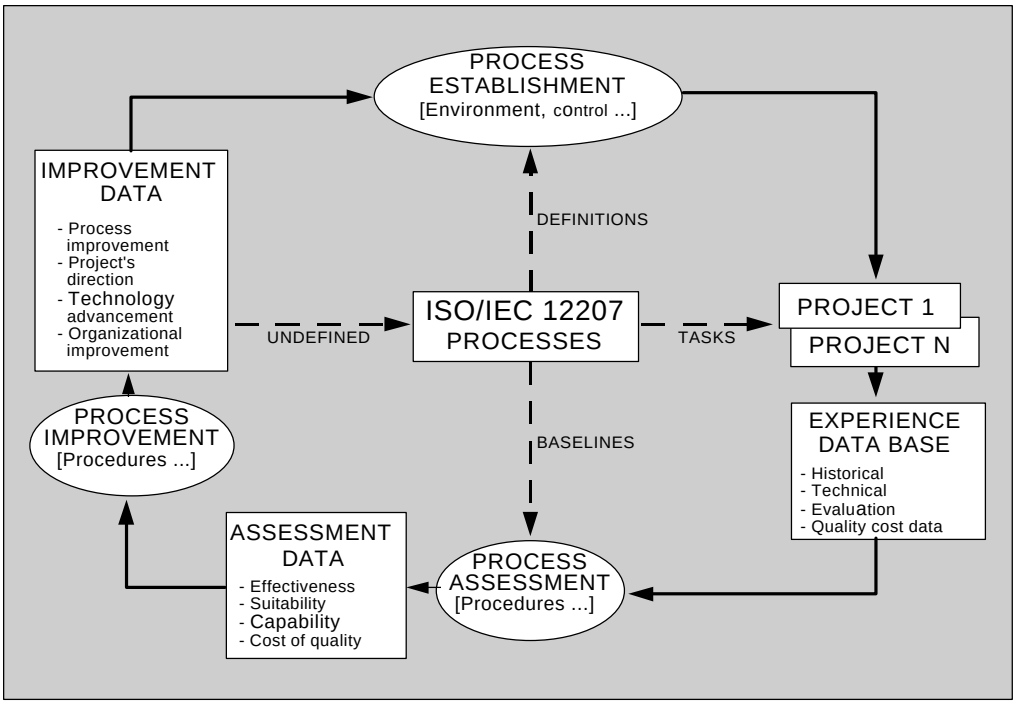

Figure 5. The Process Improvement Process

Mockup and Prototype guide. A short guide on mockup and prototype is under development by the JTC1. The guide would help in developing mockups and prototypes for software projects. It would be helpful to those projects which plan to use incremental developmental paradigms particularly. Especially, the guide would be an excellent tool for analyzing and developing requirements and design.

Software Maintenance standard. The JTC1 has approved a project for developing a standard on software maintenance. This standard, in support of ISO/IEC 12207, would provide detailed requirements and guidelines on software maintenance.

The ISO/IEC 12220 series. A series of products are being developed or planned to provide additional detailed guidelines on the supporting processes of ISO/IEC 12207. These products are listed below:

(1) ISO/IEC 12220-1: Overview Document;
(2) ISO/IEC 12220-2: Software Configuration Management;
(3) ISO/IEC 12220-3: Software Project Management;
(4) ISO/IEC 12220-4: Software Quality Assurance;
(5) ISO/IEC 12220-5: Software Verification and Validation;
(6) ISO/IEC 12220-6: Software Reviews and Audits.

Software Process Assessment standards. In January 1993, the JTC1 approved the development of a set of standards on assessing software processes. The standards would provide a framework for the assessment of software processes. The framework would be employed by organizations to plan, manage, monitor, control, and improve the acquisition, supply, development, operation, and maintenance of software. In June 1993, a SPICE (Software Process Improvement and Capability dEtermination) Project was established to undertake the development of those standards and create market awareness of them.

Drafts of these standards were produced, and these standards are under trial use currently and still evolving. The SPICE-project standards are aligned with ISO/IEC 12207.

Standard on System Life Cycle Processes. The JTC1 is developing an International Standard on System Life Cycle Processes [4]. Since the advent of software on the scientific and business fields, software engineering and hardware engineering have been evolving separately in their own fashions. As software performs increasingly critical functions in systems, harmonization and integration of software engineering and hardware engineering become a necessity. Hopefully, the future standard on system life cycle would provide a standard framework for addressing hardware, software, and human-machine interfaces in a concurrent, coherent manner.

Relationship with ISO 9001 [5]. ISO/IEC 12207 invokes ISO 9001 for additional quality assurance functions beyond those required in the Quality Assurance Process. Simultaneous applications of these two standards should be fairly non-conflicting. However, there are certain differences, which the users of the standards should be aware of before applying them on projects. These differences are summarized as follows:

(1) ISO 9001 is applicable to general systems.

ISO/IEC 12207 is applicable to general software products and services, but within the context of a system.

(2) ISO 9001 is primarily oriented to the production and manufacturing of hardware and oblivious of software characteristics.

ISO/IEC 12207 focuses on the development, operation, and maintenance of software with the necessary links to the parent system.

(3) ISO 9001 presents a corporate view of quality.

ISO/IEC 12207 presents a functional view of life cycle.

(4) ISO 9001 consolidates the activities related to quality system. It does not fully cover management and technical activities related to system engineering or software engineering.

ISO/IEC 12207 consolidates all known activities in a life cycle, including management, technical, and quality ensurance and assurance.

The terms "design/development" do not mean the same in ISO 9001 and ISO/IEC 12207. The differences, however, are not precisely clear.

(5) ISO 9001 addresses compliance with general process(es).

ISO/IEC 12207 addresses compliance with applicable processes.

(6) ISO 9001 is primarily used in determining the capability of a supplier's quality system.

ISO/IEC 12207 is primarily used for acquiring, supplying, developing, operating, and maintaining software. Assessment procedures for determining an acquirer's or a supplier's capability with respect to ISO/IEC 12207 are not available.

Relationships among ISO 9001, SPICE and ISO/IEC 12207. At the process level, the ISO, IEC and JTC1 have been active in the standardization of quality assurance, process assessment, and life cycle processes. The former has been accomplished at the system level; the latter two are being pursued at the software level. Figure 6 [2] depicts the relationships among quality assurance, process assessment, and life cycle processes standardizations. At the top left corner, ISO 9001 represents quality assurance, albeit at the system level. At the top right corner, SPICE represents process assessment as applied in organizations. At the bottom center, ISO/IEC 12207 represents the processes employed throughout the life cycle of a software product. ISO 9001, as the arrows from it show, provides the basis for quality assurance to both ISO/IEC 12207 and SPICE. ISO/IEC 12207 provides (see the arrow from it to SPICE) the baselines of life cycle processes to the SPICE assessment process. It is expected that as these standards evolve in the future, their mutual consistencies would improve.

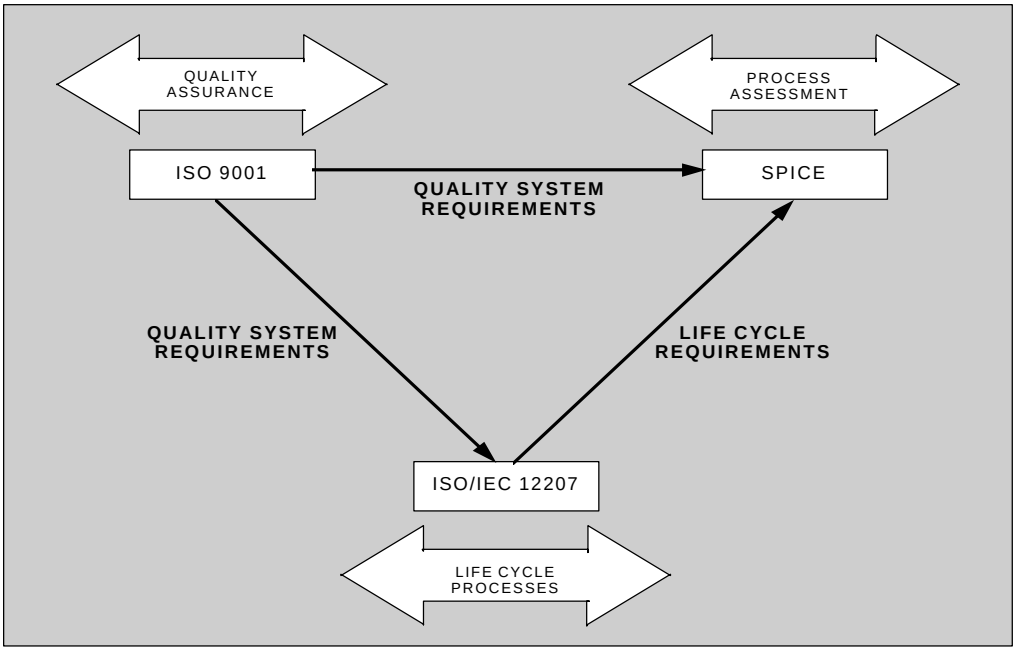

Figure 6. Relationships among ISO 9001, SPICE and ISO/IEC 12207

SUMMARY

ISO/IEC 12207 is the first International Standard that provides a complete set of processes for acquiring and supplying software products and services. These processes may be employed also to manage, engineer, use, and improve software throughout its life cycle. Its architecture can accommodate evolving, modern software methods, techniques, tools, and engineering environments. It is expected that ISO/IEC 12207 would fulfill its intended purpose as the basis for World trade in software products and services.

REFERENCES

1. ISO/IEC 12207: 1995, Information Technology - Software life cycle processes.

2. Raghu Singh; An introduction to ISO/IEC 12207 (Tutorial), August 1995.

3. ISO/IEC 9126: 1991, Information Technology - Software product evaluation - Quality characteristics and guidelines for their use.

4. ISO/IEC JTC1/SC7 N1331: 1995, Report of the JTC1/SC7 ad hoc Study Team on software-system relationships.

5. ISO 9001: 1994, Quality systems - Models for quality assurance in design/development, production, installation and servicing.