INTERNATIONAL STANDARD

ISO/IEC/ IEEE 24748-5

First edition
2017-06

Systems and software engineering — Life cycle management —

Part 5: Software development planning

Ingénierie des systèmes et du logiciel — Gestion du cycle de vie — Partie 5: Planification de développement de logiciel

Reference number
ISO/IEC/IEEE 24748-5:2017(E)

Contents

Foreword

Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and nongovernmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

Attention is called to the possibility that implementation of this standard may require the use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. ISO/IEEE is not responsible for identifying essential patents or patent claims for which a license may be required, for conducting inquiries into the legal validity or scope of patents or patent claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance or a Patent Statement and Licensing Declaration Form, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from ISO or the IEEE Standards Association.

Use of IEEE Standards documents is wholly voluntary. IEEE documents are made available for use subject to important notices and legal disclaimers (see http://standards.ieee.org/IPR/disclaimers.html for more information).

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information Technology*, Subcommittee SC 7, *Systems and software engineering,* in cooperation with IEEE Computer Society Systems and Software Engineering Standards Committee, under the Partner Standards Development Organization cooperation agreement between ISO and IEEE.

A list of all parts in the ISO/IEC/IEEE 24748 series can be found on the ISO website.

Introduction

ISO/IEC/IEEE 24748 provides unified and consolidated guidance on the life cycle management of systems and software. This document draws on key aspects of the former IEEE J-Std-016 *Standard for information technology software — Software life cycle processes — Software development — Acquirer-supplier agreement*. The IEEE has identified the need for a non-military standard to guide managers of software systems in software development planning.

Taken together, the parts of ISO/IEC/IEEE 24748 are intended to facilitate the joint usage of the process content of ISO/IEC/IEEE FDIS 12207 *Systems and software engineering — Software life cycle processes* and ISO/IEC/IEEE 15288, *Systems and software engineering — System life cycle processes*, which in turn may be used together with related standards, such as for Information Technology (IT) service management and various lower-level process standards.

The acquisition or supply of a software system is usually done within a project. A project prepares and implements the technical plans and schedules necessary to guide the project toward accomplishment of its objectives and proper conclusion. Given the project's authorization and objectives, the project should establish plans for the technical management of activities as necessary for the software development effort.

This document unifies technical and management requirements and guidance from several sources to specify the requirements for software engineering planning, including software development plans or software engineering plans. This document also identifies the processes as defined in ISO/IEC/IEEE FDIS 12207 to perform the necessary project planning activities to accomplish the project's technical effort and to develop the software project's technical management and development plans.

This document focuses on the processes required for successful planning and management of the project's software development effort and for development of the software development plan (SDP) as a vehicle for representing a project's application of software life cycle processes. The SDP is a top level technical planning document for a project which addresses technical management processes established by three principal sources (the project's agreement, applicable organizational and technical management processes, and the software development project team) as necessary to successfully accomplish the software development related tasks of the project.

Systems and software engineering — Life cycle management — Part 5: Software development planning

1 Scope

This document provides a common framework for planning and controlling the technical processes and activities to produce and sustain software products. The complete life cycle is covered by this document, from idea conception to the retirement of a software product. The framework described by this document provides for best practices in communication and cooperation among parties that plan for, develop, utilize, and manage modern software.

This document:

— specifies the required information items to be produced through the implementation of the required planning and control processes;

— specifies the required content of the required information items;

— gives guidelines for the format and content of the required and related information items; and

— details the processes necessary to develop and implement a software plan.

This document is intended to provide guidance for parties involved in the planning of software engineering at all stages of the software life cycle. It is intended to provide a common framework for two-party and multi-party collaborations and can be applied where the parties are from the same organization. This document can also be used by a single party.

This document is applicable to:

— those who use ISO/IEC/IEEE FDIS 12207 on projects dealing with software products and services related to those products;

— those who are responsible for the technical management of the development of software systems;

— organizations and individuals performing software development activities; and

— organizations and individuals developing information items during the development of software.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC/IEEE FDIS 12207:2017[1], *Systems and software engineering — Software life cycle processes*

3 Terms, definitions and abbreviated terms

For the purposes of this document, the terms and definitions given in ISO/IEC/IEEE FDIS 12207, ISO/IEC TS 24748-1:2016, and the following apply.

For additional terms and definitions, consult ISO/IEC/IEEE 24765, available at www.computer.org/sevocab.

ISO, IEC and IEEE maintain terminological databases for use in standardization at the following addresses:

[1] Under preparation. (Stage at time of publication ISO/IEC/IEEE FDIS 12207)

— IEC Electropedia: available at http://www.electropedia.org/
— ISO Online browsing platform: available at http://www.iso.org/obp
— IEEE Standards Dictionary Online: available at http://dictionary.ieee.org

3.1
acceptance
action by an authorized representative of the acquirer by which the acquirer assumes ownership of products as partial or complete performance of an agreement

3.2
audit
independent examination of a work product or set of work products to assess compliance with specifications, standards, contractual agreements, or other criteria

Note 1 to entry: Independent assessment of software products and processes conducted by an authorized person in order to assess compliance with requirements.

[SOURCE: ISO/IEC/IEEE 15288:2015, 4.1.10]

3.3
configuration item
item or aggregation of hardware, software, or both, that is designated for configuration management and treated as a single entity in the configuration management process

[SOURCE: ISO/IEC/IEEE 15288:2015, 4.1.13]

3.4
document
uniquely identified unit of information for human use, such as a report, specification, manual or book, in printed or electronic form

[SOURCE: ISO/IEC/IEEE 15289:2015, 5.10]

3.5
estimation
process of developing a quantitative assessment of the likely amount or outcome

3.6
evaluation
systematic determination of the extent to which an entity meets its specified criteria

Note 1 to entry: The entity can be an item or activity.

[SOURCE: ISO/IEC 25001:2014, 4.1]

3.7
information item
separately identifiable body of information that is produced, stored, and delivered for human use

[SOURCE: ISO/IEC/IEEE 15289:2015, 5.13]

3.8
plan

information item that presents a systematic course of action for achieving a declared purpose, including when, where, how, and by whom specific activities are to be performed

Note 1 to entry: The plan can also state artifacts that are to be created.

Note 2 to entry: Annex A provides contents of a generic plan.

[SOURCE: ISO/IEC/IEEE 15289:2011, 5.16]

3.9
planning

activities concerned with the specification of a plan

3.10
project manager

stakeholder with overall responsibility for the planning, execution, and closure of a project

Note 1 to entry: According to ISO/IEC/IEEE FDIS 12207, the project closure is performed in the Portfolio Management process.

3.11
project management plan

information item that describes how the project will be executed, monitored, and controlled

Note 1 to entry : The plan typically describes the work to be done, the resources required, the methods to be used, the procedures to be followed, the schedules to be met, and the way that the project is organized.

[SOURCE: ISO/IEC/IEEE 24765, modified – to distinguish between the technical and management approaches. See also software development plan.]

3.12
record, verb

set down in a manner that can be retrieved and viewed

3.13
record, noun

set of related data items treated as a unit

[SOURCE: ISO/IEC/IEEE 15289:2015, 5.22]

3.14
software development environment

facilities, hardware, software, procedures, and documentation needed to perform software development

Note 1 to entry: Elements can include computer-aided software engineering (CASE) tools, compilers, assemblers, linkers, loaders, operating systems, debuggers, simulators, emulators, documentation tools, and database management systems.

Note 2 to entry: Plans for software development environments can include where the specified environment is to be constructed, when sites provide different environments or facilities. For example, different testing environments can be requested to be constructed at the acquirer's site and the supplier's site.

3.15

software development plan

SDP

information item that describes the technical approach to be followed for a software development effort

Note 1 to entry: The software development plan presents how the organization or project plans to conduct development activities. A distinction is being made between the technical and management approaches. *See also* project management plan.

3.16

software engineering

systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software

[SOURCE: ISO/IEC/IEEE 24765]

3.17

software item

item

identifiable part of a software product

EXAMPLE Source code, object code, control code, control data, or a collection of these items

Note 1 to entry: A software item can be viewed as a system element of ISO/IEC/IEEE 15288:2015.

[SOURCE: ISO/IEC/IEEE 15289:2015, 5.26]

3.18

software product

set of computer programs, procedures, and possibly associated documentation and data

3.19

software quality characteristic

category of software quality attributes that bears on software quality

[SOURCE: ISO/IEC 25000:2015, 4.52]

3.20

software system

system for which software is of primary importance to the stakeholders

3.21

software unit

atomic software component of the software architecture that can be subjected to stand-alone testing

3.22

technical manager

stakeholder with responsibility for decisions relating to product content and quality achievement

Note 1 to entry: Technical decisions include definition and tailoring of life cycle processes, design of measurement systems, and product implementation decisions

3.23
work breakdown structure
hierarchical decomposition of the total scope of work to be carried out by the project team to accomplish the project objectives and create the required deliverables

Note 1 to entry: It organizes and defines the total scope of the project.

4 Abbreviations

CI	configuration item
CM	configuration management
COTS	commercial off the shelf
ICWG	interface control working group
IT	information technology
LOC	lines of code
SDP	software development plan
SEMP	systems engineering management plan
TPM	technical performance measure
WBS	work breakdown structure

5 Conformance

5.1 General

This document may be used as a conformance or a guidance document for projects and organizations claiming conformance to ISO/IEC/IEEE FDIS 12207.

To claim conformance to this document, having tailored the selected software life cycle processes, the users of this document shall prepare the information items identified in this document applicable to the selected and tailored ISO/IEC/IEEE FDIS 12207 processes.

The generic and specific record and information item contents in Clauses 8 and 9 of this document may be tailored to satisfy requirements of an organization, its projects, or agreements based on the tailored conformance to ISO/IEC/IEEE FDIS 12207. In tailoring, information item titles and contents provided in this document may be modified (added to, combined or retitled). The contents of the information items shall correspond to the selected and tailored processes.

NOTE ISO/IEC/IEEE FDIS 12207:2017, Annex A, provides requirements for the Tailoring process.

Throughout this document, "shall" is used to express a provision that is normative, "should" to express a recommendation among other possibilities, and "may" to indicate a course of action permissible within the limits of this document.

The verb "include" used in this document indicates that either:

a) the information is present; or
b) a reference to the information is listed.

5.2 Intended usage

This document:

— specifies the required set of processes or activities that deal with the technical planning of a software development effort and that are detailed during the evolution of the software development plan, and

— provides normative definition of the content of the information items that result from the implementation of these processes.

Users of this document can claim conformance to the process provisions or to the information item provisions, or both.

The requirements in this document are contained in Clauses 6, 8, 9, and 10.

5.3 Conformance to processes

This document provides requirements for the processes to be selected from ISO/IEC/IEEE FDIS 12207, necessary for planning the technical management of software development, and suitable for usage during the life cycle of a software system or product.

The process requirements in this document are contained in Clause 9.

NOTE A claim to tailored conformance to ISO/IEC/IEEE FDIS 12207 does not necessarily imply conformance to the processes in this document. When claiming tailored conformance to the processes, the directions given in ISO/IEC/IEEE FDIS 12207:2017, 2.3, apply.

5.4 Conformance to information item content

This document provides requirements for a number of information items to be produced during the life cycle of a software system or product.

In this document, for simplicity of reference, each information item is described as if it were published as a separate document. However, information items shall be considered as conforming if they are unpublished but available in a repository for reference, divided into separate documents or volumes, or combined with other information items into one document. Use of the nomenclature of the specific records or the information item titles is not required to claim conformance with this document.

A claim of conformance to the information item provisions of this document means that:

— the required information items stated in this document are produced; and

— the information items produced during the processes demonstrate conformity to the content requirements defined in this document.

The requirements for the content of the information items in this document are contained in 6.6.

NOTE If a user of this document claims full conformance to ISO/IEC/IEEE 15289, it does not imply that the user can claim conformance to the information items and information item content in this document. The reasons for this are:

1) ISO/IEC/IEEE 15289 does not contain requirements for all the specific information items listed in this document; and

2) ISO/IEC/IEEE 15289 does not contain normative requirements for all the content of the information items listed in this document.

5.5 Full conformance

A claim of full conformance to this document is the equivalent of claiming conformance to all of the requirements ("shall" statements).

5.6 Tailored conformance

When this document is used as a basis for establishing a set of information items that do not qualify for full conformance, the clauses of this document are selected or modified in accordance with the tailoring process prescribed in ISO/IEC/IEEE FDIS 12207:2017, Annex A. The tailored text, for which tailored conformance is claimed, shall be declared. Tailored conformance is achieved by demonstrating that requirements for the information items, as tailored, have been satisfied using the outcomes of the Tailoring process as evidence.

6 Concepts

6.1 General

This clause presents essential concepts on which this document is based. These concepts apply to, and are necessary for, understanding the technical planning of a software development effort, as well as the related information items that are to be produced.

NOTE Annex C provides guidance on how related standards discuss these concepts.

6.2 System concepts

Technical planning of a software development effort assumes an understanding of system concepts. This document is intended to provide guidance for the key stakeholders involved in designing, developing, and delivering systems.

The systems described in this document are designed and developed for the benefit of users, acquirers, and other stakeholders. A system is delivered to solve a problem or provide a service described by the acquirer, and it meets a set of requirements customized to the acquirer's needs. It may include software, hardware, data, processes, materials, and naturally occurring entities.

A system is designed and its deliverables developed and tested. Technical planning accounts for all of the activities, tasks, and efforts required to develop a system that meets a set of requirements.

NOTE System and software concepts are introduced in ISO/IEC/IEEE FDIS 12207:2017, 5.2. Additional discussion, such as systems and system structure, is provided in ISO/IEC TS 24748-1:2016, 3.1.

6.3 Life cycle concepts

Technical planning of a software development effort assumes an understanding of life cycle concepts.

Every software system has a life cycle. A life cycle can be described using an abstract functional model that represents the conceptualization of a need for the system, its realization, utilization, evolution and disposal.

Technical planning takes into consideration the entire life cycle of a system, from conception to retirement. This document provides guidance for the initial design and development effort, often the most laborious component of planning. However, this document also provides guidance for further efforts, including iterative improvements to a system and retirement of a system.

NOTE 1 Life cycle concepts are introduced in ISO/IEC/IEEE FDIS 12207:2017, 5.4. Additional discussion is in ISO/IEC TR 24748-3.

NOTE 2 INCOSE Systems Engineering Handbook discusses system life cycle concepts in terms of business, budget and technical aspects, and project cycles in terms of decision gates. Discussion of different methods, implementation strategies and case studies highlight some of decisions facing organizations and projects in determining appropriate system and life cycle models to employ.

6.4 Process concepts

Technical planning for a software development effort assumes an understanding of process concepts. Technical planning includes the planning for and evaluation of the processes that an organization employs to meet the requirements specified for a system.

This document provides requirements and guidance for the processes that can be performed during the life cycle of a software system. Effective processes should be recommended to all stakeholders involved in the development effort. Stakeholders adhere to the accepted processes to increase the efficiency and effectiveness of the development effort.

The Software Development Plan (SDP) acts as a repository for processes specific to a project or organization. Other processes may be introduced external to the SDP, as part of the culture and Standard Operating Procedure of the organization.

NOTE 1 Process concepts are introduced in ISO/IEC/IEEE FDIS 12207:2017, 5.5 and ISO/IEC TS 24748-1:2016, 3.3.

NOTE 2 Process concepts as related to software are discussed in ISO/IEC TR 24748-3:2011, 4.4.

6.5 Project concepts

Technical planning of a software development effort assumes an understanding of project concepts. As defined in ISO/IEC/IEEE FDIS 12207, a project is an endeavor with defined start and finish criteria undertaken to create a product or service in accordance with specified resources and requirements, while a project portfolio is a collection of projects that addresses the strategic objectives of the organization.

A project may be viewed as a unique process comprising coordinated and controlled activities and may be composed of activities from the Technical Management processes and Technical processes defined in ISO/IEC/IEEE FDIS 12207. In ISO/IEC/IEEE FDIS 12207 and in this document, the project has been chosen as the context for describing processes concerned with planning, assessment and control.

There need not be a one-to-one relationship between a software development effort and the projects related to the effort. However, for the purpose of this document, software development is done within a project.

NOTE 1 Project concepts are discussed in ISO/IEC/IEEE TR 24748-3:2011, 4.6.

NOTE 2 ISO/IEC TS 24748-1:2016, 3.1.4, provides more detail on structure in systems and projects, and 3.1.5 provides more detail on enabling systems.

NOTE 3 ISO/IEC/IEEE 16326 provides more information on project management and the project management plan.

NOTE 4 Software Extension to the PMBOK® Guide, 5th edition, provides more information on projects and project management.

6.6 Information item concepts

6.6.1 General

ISO/IEC/IEEE FDIS 12207 defines an information management process, but does not detail documentation in terms of name, format, explicit content, and recording media. ISO/IEC/IEEE FDIS 12207 establishes a common framework for software life-cycle processes and requires a number of information items. It does not always specify when software information items are to be prepared, nor does it identify information item contents.

NOTE 1 ISO/IEC/IEEE 15289 provides a mapping of ISO/IEC/IEEE FDIS 12207 clauses with a set of information items. It specifies how life cycle data is managed in information items. ISO/IEC/IEEE 15289:2015, 6.1, provides requirements for life cycle data characteristics.

For each life-cycle process, it would be possible to prepare a plan, procedures, and reports, as well as numerous records, requests, descriptions and specifications. Such an elaboration of the documentation schema would be more rigorous than specified by ISO/IEC/IEEE FDIS 12207. ISO/IEC/IEEE FDIS 12207 does not detail the life-cycle processes in terms of methods or procedures required to meet the requirements and outcomes of a process. Thus, information items may be combined or subdivided as needed for project or organizational purposes.

Information item content is information included in an information item, associated with a system, product or service, to satisfy a requirement or need.

An information item has to be consistent with an information item generic type. An information item type is a group of information items consistent with a pre-arranged set of generic criteria.

NOTE 2 ISO/IEC/IEEE 15289:2015, Clause 7, provides more information on contents of generic information types.

ISO/IEC/IEEE 15289 makes a distinction between records and documents (which includes plans). Each information item produced as a document supports certain life cycle data characteristics. A document is a generic term for separately identifiable, published (in electronic or printed media) information. Documents are produced and communicated for human use and contain formal elements (such as purpose, scope, and summary), intended to make them usable by their intended audience.

In this document, for simplicity of reference, each information item is described as if it was published as a single document. However, the information items may be considered as conforming if it is unpublished but available in a repository for reference, divided into separate documents or volumes, or combined with other information items into one document.

6.6.2 Concept of Plans

A Systems Engineering Management Plan (SEMP) is a top-level technical planning document for a project which addresses technical management processes as necessary to successfully accomplish the systems engineering-related tasks of the project. The technical management processes are typically established by three principal sources:

1) the project’s contract or agreement;
2) applicable organizational processes; and
3) the systems engineering project team.

NOTE Annex A provides guidance for a generic plan.

6.6.3 Software development plans

The plan addressed in this document is often called the Software Development Plan (SDP), Software Engineering Management Plan, or Software Engineering Plan. The SDP presents how the organization or project plans to conduct software development or software engineering activities (the software implementation strategy). It is a comprehensive, composite artifact that contains all information required to manage the software development activities.

The SDP itself, coupled with the process of developing it, helps an organization produce a software system in a more structured, predictable, and efficient way. It provides documented confirmation of project details, which helps ensure a common understanding of the project among a varied group of stakeholders. It helps an organization to estimate time and resources required to develop the project from specifications. Finally, the SDP also contributes to clarity of roles and responsibilities and establishes continuity in the event of changes to project requirements.

NOTE The required content of the SDP is in 9.2; Annex B provides an example outline of an SDP.

6.7 Management concepts

6.7.1 General

This document describes two management roles: project and technical. Technical management and project management are collaborative efforts that, taken together, manage a software development project. The role of a project manager is distinct from that of a technical manager. For the purpose of this document, the project manager is responsible for all tasks related to project management, including scheduling and resource allocation. Organizational management provides project managers with clear goals, obvious performance measures, resources for assigned tasks, and time for decision making. The technical manager takes responsibility for technical aspects of the project in support of the project manager. The document does not specify the assignment of roles to specific persons, or enforce a specific structure on the project or development teams. The same role can be filled by more than one person (e.g., a manager and a deputy manager) and one person can be both the project and the technical manager, especially on small projects.

6.7.2 Project management

Project management concepts are used to establish and evolve project plans, to assess actual achievement and progress against the plans and to control execution of the project through to fulfilment. Various project management processes may be invoked at any time in the life cycle and at any level in a hierarchy of projects.

Project management is responsible for resources, schedules, stakeholder management, and achievement of stakeholder concerns other than technical concerns, such as profitability and marketability.

Knowledge created on projects supports the organization’s capability and assets that enable the organization to exploit opportunities.

NOTE ISO/IEC/IEEE 16326:2009 provides more detail on project knowledge management.

6.7.3 Technical management

The technical manager is responsible for decisions relating to product content and quality achievement, including definition and tailoring of life cycle processes, design of measurement systems, and product technical decisions. Product technical decisions may include practices for requirements, architecture and design, definition of the methods for development, validation, integration and deployment, and overall achievement of quality. The technical and project manager are both responsible for risk management. Project management is responsible for project risk management, and technical management is responsible for technical risk management.

Technical management responsibilities include the identification and performance of the activities needed to produce the product, ensuring the content of each technical information item and ensuring mutual consistency among the information items.

6.8 Software development model concepts

6.8.1 General

This document is designed to accommodate any software development methodology. The technical manager chooses a methodology appropriate to both the project and the organization. Often, an organization employs the same software development model for most or all projects it undertakes.

An organization can claim conformance to this document within the context of any software development model it uses. Moreover, various development methodologies, including commonly used Agile methods, may be employed within the context of any of the models described below.

Although innumerable distinct models exist, most fall within one or more of the following types:

— Once-Through (also known as Predictive or “Waterfall”);

— Evolutionary; and
— Incremental.

6.8.2 Once-Through software development model

Using the Once-Through software development model, the development process is performed a single time for a project. Requirements are defined, the software is designed and developed, the software is tested and integrated, and finally it is released.

Once the software is released, the software is understood to be completed and the project rendered inactive. When updates are required to the software, a new and separate software development project can be undertaken.

In the Once-Through model, the SDP created at the beginning of the project is understood by all parties to be static. Modifications to the SDP may be introduced by the technical manager in certain situations, including but not limited to the following:

— significant issues arise during the testing phase, requiring software to be significantly redeveloped;
— major changes to software requirements are instituted by key stakeholders;
— personnel issues require developers and others to be added to or removed from the project;
— internal or external deadlines are missed or significantly changed; and
— significant budget overruns have occurred or allocated resources have been reduced.

6.8.3 Evolutionary software development model

The Evolutionary software development model develops software over the course of several builds (also called "iterations"). Requirements are partially defined at the start of the project but are understood to evolve over the life cycle of the software.

Unlike a Once-Through project, which is planned and scheduled in its entirety from the beginning, an Evolutionary project is planned and scheduled in detail for only one iteration at a time. Requirements are specified at the beginning of each build. These requirements are then developed in small increments, tested, and revised if necessary. The processes of defining requirements, developing, testing, and revising software all happen concurrently.

Once software is released, the development process continues. Input from users shapes the software over time, and the project is not considered completed at the first release.

In the Evolutionary model, the SDP created at the beginning of the project is understood to be incomplete. The stakeholders agree on high-level requirements, a release date, and development approach. At the beginning of the project, the requirements for the first iteration are defined in some detail, whereas later iterations are specified far more loosely. In particular, a schedule template, without technical requirements, is defined for later iterations.

The SDP defines the overall timeline for the development cycle and may predetermine the order in which requirements are developed.

The SDP complements the individual requirements that are created in this model. The technical manager oversees the person responsible for developing requirements. The requirements are based on the current draft of the SDP, the requirements as currently understood, and the constraints facing the developers.

Technical review meetings serve as forums to evaluate updates to the SDP, and these changes can be approved by the acquirer at regular status meetings. Some changes do not require formal acceptance by the acquirer.

6.8.4 Incremental software development model

The Incremental software development model is a joining of Once-Through and Evolutionary models. Like the Once-Through model, requirements are specified as fully as possible at the beginning of the project. Like the Evolutionary model, development happens over the course of several builds.

The requirements are defined early in the software life cycle and are distributed among multiple builds. The first build incorporates part of the planned capabilities; the next build adds more capabilities, and so on, until the software is complete.

In the Incremental model, a single SDP represents all builds. The SDP created at the beginning of the project is understood to be as complete as possible. The first build is described in the greatest detail, and subsequent builds are described with correspondingly less detail.

Although the high-level requirements are expected to remain static, in this model the schedule information is modified and updated regularly to reflect a changing project, release dates, and resources needed as the software development team progresses from build to build. This may be maintained in a separate information item that does not require formal acceptance by the acquirer.

At the beginning of each build, the SDP is reviewed by the key stakeholders. Updates to the SDP are made only when they relate to the current build; future builds are not reviewed or updated in the SDP unless requirements change.

7 Software life cycle processes and software development planning

7.1 General

ISO/IEC/IEEE FDIS 12207 requires a number of processes for the development of a software system. These processes are influenced by, or influence, the technical management in a project. These processes are presented in four process groups:

1) Agreement processes;
2) Organizational project-enabling processes;
3) Technical management processes; and
4) Technical processes.

7.2 Agreement processes

As stated in ISO/IEC/IEEE FDIS 12207, the Agreement processes define the activities necessary to establish an agreement between two organizations. It follows that these processes have a significant influence on the project and planning processes. They place requirements and constraints on all planning aspects. The Agreement processes are out of the scope of this document; however, the agreement itself may specify methods and constraints that affect the SDP and the project schedule.

NOTE The Agreement processes are discussed in ISO/IEC/IEEE FDIS 12207:2017, 6.1.

7.3 Organizational project-enabling processes

As stated in ISO/IEC/IEEE FDIS 12207, the Organizational project-enabling processes manage the organization’s capability to acquire and supply products or services through the initiation, support and control of projects. These processes influence the technical management of a project and place constraints on the resources and infrastructure required to support a project.

The Organizational project-enabling processes support the project and activities described in this document.

Planning may also influence the Organizational project-enabling processes to make provisions for required infrastructure and resources. Providing infrastructure and resources that support project objectives is a key part of software development planning. Existing infrastructure may need to be modified or new resources added in order to realize a successful software project.

NOTE 1 The Organizational project-enabling processes are discussed in ISO/IEC/IEEE FDIS 12207:2017, 6.2.

NOTE 2 Infrastructure-management processes are discussed in ISO/IEC/IEEE FDIS 12207:2017, 6.2.2.

7.4 Technical management processes

Project planning and technical planning are closely related, with technical planning mostly in support of the project planning, especially contributing to planning for resource requirements, as well as the influence on time duration due to technical issues.

In particular, the Technical Management processes include project planning processes and project assessment and control processes, which are essential activities in software development planning as discussed throughout this document.

The Technical Management processes are applicable to this document, as discussed in subsequent clauses.

NOTE The Technical Management processes are discussed in ISO/IEC/IEEE FDIS 12207:2017, 6.3.

7.5 Technical processes

The Technical processes are used to define the requirements for a system, to transform the requirements into an effective software system or product, and to dispose of the product when it is retired from service.

NOTE The Technical processes are discussed in ISO/IEC/IEEE FDIS 12207:2017, 6.4.

8 Software development planning

Software development planning shall be conducted to enable the software system development project to implement the following technical management processes as defined in ISO/IEC/IEEE FDIS 12207:

— Project planning (ISO/IEC/IEEE FDIS 12207:2017, 6.3.1)
— Project assessment and control (ISO/IEC/IEEE FDIS 12207:2017, 6.3.2)
— Decision management (ISO/IEC/IEEE FDIS 12207:2017, 6.3.3)
— Risk management (ISO/IEC/IEEE FDIS 12207:2017, 6.3.4)
— Configuration management (ISO/IEC/IEEE FDIS 12207:2017, 6.3.5)
— Information management (ISO/IEC/IEEE 12207, 6.3.6)
— Measurement (ISO/IEC/IEEE FDIS 12207:2017, 6.3.7)
— Quality assurance (ISO/IEC/IEEE FDIS 12207:2017, 6.3.8)

The project shall define a software development strategy.

The software development strategy should include the following:

a) development methods, including software development, programming or coding standards, unit test policies, and language-specific standards for implementing security features;
b) implementation procedures for software development and development of unit tests;
c) the use of peer reviews, unit tests, and walkthroughs during implementation;
d) activities under CM control during software construction;
e) change management considerations for manual processes;
f) implementation priorities to support data and software migration and transition, along with retirement of legacy systems; and
g) creation of test procedures to verify that a software unit meets its requirements along with creation of the software unit.

NOTE Software implementation strategy is described in ISO/IEC/IEEE FDIS 12207:2017, 6.4.7.3.a)1).

9 Process Execution

9.1 Overview

The software development technical management processes, activities, and tasks are found in ISO/IEC/IEEE FDIS 12207, Software life cycle processes. This document elaborates on these processes to provide the user with additional planning and implementation requirements and guidance specific to software development planning.

Acquisition and supply processes are not in scope of this document, since they and the produced plans are not considered as part of the technical planning for a project that occurs during the development of the software.

In this clause, ISO/IEC/IEEE FDIS 12207 clauses relevant to this document are referenced as primary source materials.

The original ISO/IEC/IEEE FDIS 12207 purposes and outcomes relevant to this document are used in their entirety, without any change, for the subset of processes that are relevant to software development technical management processes.

A number of these tasks can be executed as part of project management, but this does not detract from their requirement as part of development planning or technical management. This document provides guidance from a development planning and technical management point of view.

NOTE Although the applicable processes are stated as normative, it should be stressed that the guidance provided below assumes knowledge of the purpose, outcomes, and activities of each process, and should be read in conjunction with these process aspects.

9.2 Project planning process

9.2.1 General

The purpose of the Project planning process is to produce and communicate effective and workable project plans. This process determines the scope of the technical activities.

If a project is developed in multiple builds, then planning for each build may result in multiple project plans. These build-specific project plans include:

a) context of the current build in the overall project,
b) detailed planning for the current build,
c) planning for future builds.

The technical plans addressed in this document form a subset of the project plans. Moreover, project planning and the associated plans are dependent on input from the technical manager and the technical planning.

NOTE ISO/IEC/IEEE FDIS 12207:2017, 6.3.1, provides activities and tasks related to the project planning process.

9.2.2 Responsibility for planning

Development planning and technical management support contribute to the outcomes and activities of the Project Planning process. The responsibility for preparing and approving plans, including technical management plans, shall be assigned and recorded.

The technical manager shall plan and execute requirements management, so that requirements related to the technical effort are elicited, documented, analyzed and managed.

The technical manager shall plan for the required software engineering resources and skills.

Software development planning includes planning for retirement. Software may be retired for several reasons, including the following:

— it is replaced by another software project;
— it has become too costly to maintain;
— it has become redundant; and
— it has become obsolete.

In the SDP, the technical manager shall identify the probable circumstances that would result in software retirement. When a new project or iteration is to be undertaken, the technical manager shall consider whether the project involves retirement of software. For example, the beginning of a new project may subsume an existing release, causing that existing release to be retired. Similarly, at the beginning of a new iteration, it may be determined that so much maintenance and code refactoring is required that the software should be retired and replaced with something new.

Technical planning should include the following activities:

a) Involving identified stakeholders in the requirements definition.
b) Managing change to the scope and requirements throughout a project's life cycle. Changes in scope and requirements should be evaluated for the impact on cost, schedule, risk, current scope and quality.
c) Reviewing the selection of processes when the scope and requirements are changed, to assure that the selected processes are still applicable after the changes in scope and requirements.
d) Defining who is responsible for obtaining stakeholder agreement on requirements.
e) Establishing and maintaining requirements traceability between levels of requirements, between requirements and design, between design and the software that implements it, between requirements and qualification test information, and between computer hardware resource utilization requirements and measured computer hardware resource utilization.

NOTE 1 ISO/IEC/IEEE 29148:2011 provides requirements and guidance on requirements engineering.

NOTE 2 ISO/IEC/IEEE TR 24748-3 provides more guidance on life cycle models.

The development and recording of planning and engineering information is an integral part of the software development process, and should be performed regardless of whether a deliverable artifact is required.

The project manager should select a life cycle model appropriate to the project. The technical manager should plan for system life cycle considerations.

9.2.3 Project scope

The technical manager shall provide technical input to support the project manager in determining the feasibility of the planned Technical Management and Technical Processes so that personnel, materials, facilities, software development environment, and technology required to execute and manage the project are available, adequate, and appropriate. The technical manager shall estimate the completion date of the software product to specifications given relevant resources and constraints. The results from these feasibility analyses could cause an adjustment to the initial project scope statement.

The technical manager shall review the project scope statement at intervals specified in the SDP to help ensure that the process is consistent with the user needs and requirements and adheres to the plans. Updates to plans should be subject to the approval of the project manager.

Technical management contributes to the development of a project scope statement. The following guidance is useful to this effect:

a) The project manager obtains from the technical manager a description of the project technical activities. These activities include how stakeholder requirements are realized in deliverables and how products are delivered as specified, i.e., verify that a project includes all the work required, and only the work required, to complete the project and product successfully.
b) The technical manager works with the project manager to confirm that process tailoring and project planning account for the type of project, systems and technologies expected to be encountered during the development.

NOTE 1 Refer to ISO/IEC/IEEE FDIS 12207:2017, Annex A, for details of the tailoring process for projects and systems expected to be utilized during development. The project manager is responsible for recording items such as the project schedule, budget, and resources.

NOTE 2 Refer to ISO/IEC TR 24748-1 and ISO/IEC 24748-3 for application guidance in tailoring and adapting the processes of ISO/IEC/IEEE FDIS 12207 for a project.

An initial scope statement is usually based on solicited and unsolicited requirements. Factors such as changes in stakeholder requirements, environment, project budget and schedule can have significant impact on scope. An evolving design may require reassessing and reaffirming agreements and commitments, and may require appropriate changes to the project scope statement.

The project scope statement, the Master Schedule, Work Breakdown Structure (WBS), and infrastructure should be the responsibility of the project manager. See 6.7 for an explanation of the distinction between the responsibilities of the project manager and the technical manager.

Planning the scope of the project may be difficult when a new project has unprecedented elements. For such a project, care should be taken to help ensure it is properly scoped and monitored. The organization's risk management process should identify detailed mitigation plans, if it is determined that additional risk exists due to the unfamiliar project. Often specialty engineering studies or activities are conducted to identify or help control project risks in unprecedented situations.

9.2.4 Work Breakdown Structure

Employing a WBS enables a project manager and technical manager to break down a large, complex project into smaller project pieces. These small components then provide a framework for organizing and managing the project and provide visibility into processes and products.

The WBS should be constructed to allow a project to be managed at the appropriate level of granularity consistent with the size, complexity, criticality, and risk of the project. It can break down the project by timeline, by phase, by function, or by system element.

The nature of the WBS differs depending on the software development model employed by the project team. In the once-through software development model, the WBS is organized around the major deliverables and software elements produced as part of the project. Project requirements are known at the beginning of a Once-Through project, and the WBS elements are estimated based on those requirements. In the evolutionary software development model, the WBS is organized around small units of functionality (e.g., user stories). For evolutionary projects, the WBS decomposes evolutionary features into epics and further into user stories.

9.2.5 Project estimation

9.2.5.1 General

The project shall use available historical project and technical data when developing estimates and plans. Mechanisms to collect, analyze, archive and retrieve project and technical data shall be developed. These historical data may be used to improve life cycle processes and support planning and analysis for future projects.

The application of historical data is relevant for both previous projects and previous iterations of a single project. If a project is developed over several iterations, the software items of each build may be refinements of, or additions to, software items of previous builds.

During project planning, multiple estimation techniques should be used. Project estimates used in software development planning should address needs of the technical effort and include:

a) nonrecurring costs to produce products and enabling systems;
b) infrastructure;
c) need for resources, including related management and control;
d) skill, experience level, and competency of assigned personnel;
e) quality assurance and control;
f) risk management;
g) software development environment provisions;
h) work to be performed in each process and/or activity;
i) configuration management; and
j) technical performance measures (e.g., response time, throughput, memory utilization, bandwidth).

Through the project estimation process, stakeholders assess the human skill, and infrastructure needs of the software project. These estimates, from software developers, quality assurance analysts, designers, and others involved in the project, are used in the SDP to predict what is required to successfully complete the project.

The project estimate can be expressed in terms of effort, relative complexity, or cost. Estimation is appropriate for any software development model and for new projects as well as improvements or maintenance on existing systems.

These estimates may be used as input for the initial SDP as well as its revised versions, pricing, and agreements. The following estimation information items are valuable to include in the SDP.

9.2.5.2 Effort and complexity estimation

This estimation task involves predicting the effort, measured in hours or a measure of complexity such as function points or story points, that are needed to produce the project to specification.

NOTE ISO/IEC 20926:2009, *Software and systems engineering — Software measurement — IFPUG functional size measurement method*, explains a common method for function point estimation.

The effort estimation shall include the time beyond what is required for constructing and integrating the software. In fact, coding is often the smallest part of the overall effort. This estimate shall account for defining requirements, architecture and design, developing and reviewing user documentation, and reviewing and testing the software. These activities often take up the larger portion of overall project effort.

The effort estimation is the sum of all the planned activities, usually plus an allowance for contingencies.

The way that effort is estimated varies in different software development models. In a Once-Through development model, effort is often measured in hours required of all internal project participants, including developers, quality assurance analysts, designers, managers. It does not usually include the hours required of external stakeholders. In an evolutionary development model, the effort is usually measured in points, with each user story assigned a specific number of points at the start of each iteration. In the evolutionary model, it is understood that the effort required for the entire project may be unquantified, although the project duration is established.

9.2.5.3 Calendar time estimation

As part of the estimation process, the technical manager predicts the time until the project is completed.

Generating an accurate schedule involves estimating the number of people to work on the project, what they will work on (based on the WBS), when they start working on the project, and when they are temporarily or permanently moved off the project, such as an expected leave or reassignment to another task. Once this information is gathered, a calendar schedule can be estimated. Historical data from the organization's past projects is often the best predictor for the number of people needed for a project of a given size.

In a once-through software development model, the schedule estimate is specifically defined. In an evolutionary model, the schedule estimate is developed for each iteration while the final completion date is dependent on user needs.

9.2.5.4 Cost estimation

The technical manager shall estimate the project cost with due consideration to other costs that may be allocated to the project, such as the time of stakeholders contributing to the project and their approximate hourly rates; the external subcontractors contributing to the project and the price for their services; the cost of capital equipment required for the project; and the price of services such as external hosting and communication providers.

NOTE Project managers, as advised by the technical manager, use existing organizational infrastructure as feasible. When existing infrastructure is inappropriate or insufficient to support a project, then adaptation or additions to existing infrastructure can be supplied by subcontractors, acquired, or developed. ISO/IEC/IEEE FDIS 12207:2017, 6.2.2, describes the infrastructure management process to support organization and project objectives throughout the software project life cycle.

9.2.5.5 Use of the software project estimates

Once the initial project estimates are complete, they should be reflected in the project budget and schedule. The technical manager shall describe the type of information used to construct the estimate in the SDP itself and shall review the estimate with key stakeholders.

The following is useful for project schedules:

a) ISO 10006 provides guidelines for project managers to help ensure proper quality of their project's products and services.

b) The planning process should establish completion criteria for all project tasks. The intent, as supported by the PMBOK Guide, 5th edition, Software Extension, is to determine if a project, activity or task has been completed successfully.

c) A software project shall have one Software Engineering Master Schedule. Subordinate schedules should be consistent with it. Often a subordinate schedule and detail schedules are used to control the software development effort and teams.

9.2.6 Relationship management

Conflict resolution and interpersonal relationships contribute to the outcomes and activities of the Project planning process. The following is useful to this effect:

a) Project planning includes a mechanism for conflict resolution and escalation so that an appropriately authorized level of organizational management may resolve disagreements among members of the technical team. Technical plans should reference this mechanism.

b) Whenever supporting processes are performed by organizations outside the direct organizational control of the project manager (or technical manager), it is important to realize the existence of two sets of relationships between:

1) the manager and the management of the outside organization; and

2) organizational management from each organization.

The project and technical managers should recognize these relationships when considering aspects of planning, implementation, control and reporting through clearly specified technical and management reporting, information flow and dispute resolution processes. Synchronization of plans may be more difficult under subcontract agreements and assignment, but can be aided by having a single master plan.

c) When multiple teams from one or more organizations participate in a project, the project manager should integrate these teams with alignment of the charter and shared vision of each team with the project's objectives.

d) The technical manager shall plan activities to resolve issues and resource needs for requirements, interfaces and design with relevant stakeholders.

e) Contingency planning for both management and technical issues shall be included in project planning.

9.2.7 Risk Management

Risk is a measure of future uncertainties in achieving program performance goals and objectives within defined cost, schedule, and performance constraints. Risk management should address risk identification, analysis, risk treatment, risk monitoring, and maintaining a risk profile. Risks should be quantified and implications reflected in the SDP. Risks will emerge and evolve throughout the duration of a project, and the SDP should be updated per guidance provided in 6.8 of this document.

9.2.8 Configuration management

The Configuration Management process allows technical insight into all levels of software design and is the principal methodology for establishing and maintaining consistency of a software product's functional, performance, and physical attributes with its requirements, design, and operational information throughout the software life cycle. Configuration Management should address identification, control, status accounting, and audits. All of these should be documented in the SDP.

9.2.9 Information management

The Information Management process concerns the acquisition of information items, the custodianship and the distribution of information items to those who need it, and its disposition via an archiving process or deletion. Information Management should address identifying relevant sources of information, specifying methods for distribution of information, and assigning criteria for information disposition and deletion. All of these should be documented in the SDP.

9.2.10 Quality Assurance

The Quality Assurance Process concerns the evaluation of a software product's effectiveness as it relates to requirements, development, and maintenance. Developing and executing a Quality Assurance plan is essential to the software development life cycle. The purpose, outcomes, and tasks of the Quality Assurance Process should be documented in the SDP.

9.3 Project and software measurement

9.3.1 Measurement overview

Measures and measurements offer insight into the effectiveness of the development process and the software itself. Throughout the course of a project, appropriate data are analyzed, compared against past averages, and evaluated. Accordingly, the SDP plans and accounts for the measures used to understand the project. The technical manager shall choose a selection of Technical Performance Measures (TPM) appropriate to the project before development begins. These are traceable to critical quality characteristics, which are derived based on the customer/user requirements during the initial phases of a project execution. The TPMs and critical quality characteristics directly influence the performance standards and selected measures for the project or product.

The measures chosen shall be included in the initial SDP, and measurements shall be taken regularly and reported to stakeholders. When measurements are taken regularly, the technical manager can compare the results to early predictions and make adjustments accordingly.

As a prediction tool, measurements can be estimated at the beginning of a project or a time interval such as the beginning of a new iteration.

EXAMPLE In a Once-Through software development model, the technical manager can predict the overall number of person-hours required by the entire project. In an Evolutionary software development model, the technical manager can predict the number of person-hours required for the next iteration.

Two kinds of measures shall be identified in the SDP: project measures and software measures. Typical project measures, described in 9.3.1, include cost, human effort, and calendar time. Typical software measures, described in 9.3.2, include lines of code, number of defects, and execution speed.

9.3.2 Project measures

Project measures assess the effectiveness of the project and its processes. These measures enable the technical manager to track potential risks, uncover problem areas, and adjust work flow or tasks.

EXAMPLE Typical project measures include schedule performance index, cost performance index, financial cost, earned value, human effort expended, and calendar time.

9.3.2.1 Cost

This measure evaluates the financial resources that have been supplied to the software project. The financial cost may include the salaries or partial salaries of permanent or contract employees hired by the organization, capital infrastructure investments, travel for meeting and testing purposes, training courses, office space, and outside resources.

The way in which a financial cost measurement is taken depends upon the organization and its methods for allocating financial and human resources. For example, some software projects measure only human resources as a contributing factor to the project's financial cost, while assuming that other costs are "overhead" absorbed by the organization.

9.3.2.2 Human effort expended

This measure evaluates how many person-hours have been contributed to the design, development, and testing of the system. This measure may also be used to estimate the financial cost, and may include every individual contributor to the project including non-technical staff.

9.3.2.3 Calendar time

In addition to the number of hours required to develop a project, elapsed calendar time is an important measure as well. Elapsed calendar time measures the total number of days and weeks from the beginning of the project to its current point.

9.3.3 Software measures

Software measures assess the effectiveness and usability of the software product that is delivered to a client or end user. These measures enable the technical manager to assess the quality of the software deliverables both before and after they are handed to a client. Performance measurements taken during development should be evaluated for comparison to the intended production environment and workload.

EXAMPLE Frequently used software measures include lines of code (LOC), execution speed, and defects per time period.

9.3.3.1 Lines of code (LOC)

The LOC measure evaluates the size of the software product given in individual lines of source code. LOC is typically used to measure the effort that has been required to develop a system. LOC is also a frequent measure of programming productivity and efficiency.

This measure is advantageous for a project because it is relatively easy to measure and quantify. For many projects, it is also a reasonable representative of the effort required. However, this measure is also disadvantageous because it is highly dependent on the programming language used. It can also be misleading relative to other measures: a more compact, efficient piece of software may take a programmer longer to design and develop than a longer piece of code.

An alternate method to measuring LOC is using function points. Function points are units of measure for the functional size of software. Using this measure helps to quantify the software deliverable based upon the user requirements.

9.3.3.2 Number of defects

The number of defects measure counts the number of distinct defects in the system. This can be measured in two phases: the number of defects uncovered before the system is released to an end-user or customer, and the number of defects uncovered by an end-user or customer.

The number of defects or software crashes can be measured within an established time period of running the software (defects per time period), or compared against an established number of lines of code in the system (defects per lines of code).

9.3.3.3 Execution speed

The response time of running an entire program is an effective measure of software efficiency. Generally, the execution speed of specific key tasks is measured, rather than the program in its entirety.

9.4 Project assessment and control

9.4.1 Overview

Part of the purpose of the Project assessment and control process is to help ensure that the project satisfies technical objectives. The objectives consider technical influences.

NOTE ISO/IEC/IEEE FDIS 12207:2017, 6.3.2, provides activities and tasks related to the Project Assessment and Control process.

9.4.2 General guidance

Development planning and technical management support contribute to the outcomes and activities of the Project Assessment and Control process. The following is useful to this effect:

a) The technical plans addressed in this document form a subset of the project plans. Project planning and the associated plans are highly dependent on input from the technical manager and the development planning.

This document supports a variety of software development models. The technical manager should define software development processes consistent with the model chosen for the project.

b) Technical working groups for the evaluation and implementation of key technical aspects of the project should be established and maintained as needed. The technical manager coordinates with working groups as specified in the SDP. Interface Control Working Groups (ICWGs) should be formed for evaluation and successful implementation of interface constraints. Each ICWG consists of a representative from each organization affected by an interface. The ICWG provide a forum to discuss software and system interfaces, explore options and reach agreement on the best approach for implementing interfaces. ICWG recommendations requiring project change should be submitted to whatever formal configuration management process the project has implemented (such as a configuration control board) for approval prior to implementation. Interfaces should be specified and controlled as an integral part of software specification using interface description documentation.

c) The technical manager verifies that the SDP contain sufficient activities to provide evidence of deviations from planned schedules and/or constraints in a timely manner that allows recovery. The project manager verifies that such deviations are accommodated in the project schedule and requested resources.

d) The technical manager is responsible for ensuring that tasks include:

   1) assessing the review results of products, activities and tasks;
   2) conforming to project and technical management plans, philosophies, methodologies and technologies;
   3) documenting plans and commitments;
   4) satisfying requirements; and
   5) determining readiness for advancement to the next process, activity or task.

e) The technical manager controls the supporting process activity that occurs in support of each project. Problems or exceptions are brought to the attention of the project manager for impact analysis on a project's cost, schedule, scope and quality.

f) Where milestones are in place and depend on achievement of technical milestones or upon reports and/or outcomes from any supporting process, the technical manager helps ensure that these achievements are reported in an accurate and timely manner in accordance with approved plans. Since it is common for project milestones to be contractually linked to the achievement of technical milestones or performance of supporting processes (e.g., the achievement of a particular baseline), it is essential that the plans are synchronized and the project manager is made aware of difficulties experienced by supporting processes in completing assigned tasks.

g) Project and technical managers shall perform periodic structured reviews of performance that are based on realistic techniques to support an accurate project assessment.

h) Recovery from a schedule slip should be carefully assessed and should be expected to impact performance, cost, risk, or quality.

i) In cooperation with stakeholders, the requirements baseline shall be reviewed regularly throughout a project to help ensure conformance with or adjustment to the objectives (cost, time and performance).

j) Managers shall specify and refine techniques to determine progress, so as to allow early detection of cost and schedule overruns. Subclause 9.3 provides additional guidance on identifying and evaluating measures that determine progress.

k) Significant issues, action items, and decisions resulting from reviews and evaluations shall be recorded. Action items and significant issues should be tracked to closure and resolution or closure of problems should be managed using a corrective action system.

l) Project and technical managers shall manage interdependencies among Technical Management processes, ensuring that dependencies are communicated to stakeholders and that relevant technical tasks are included in the SDP.

m) Changes to the scope and requirements should be managed throughout a project's life cycle. Changes in scope and requirements should be evaluated for the impact on cost, schedule, risk and quality.

n) Software items that co-occur within a project or iteration need not be completed at the same time. The project and technical manager may allow software items to be delivered separately if this technique allows for optimal scheduling.

o) The selection of processes should be reviewed when the scope and requirements are changed, to assure that the selected processes are still applicable after the changes in scope and requirements.

NOTE ISO/IEC/IEEE 29148 provides requirements and guidance on requirements engineering.

9.4.3 Project assessment and control process

The technical manager shall periodically assess the work accomplished by the team in relationship to the expectations described in the SDP and project management documents. Based on these assessments, the technical manager shall identify necessary and beneficial changes to the scope and timeline of the project, shall propose updates to the SDP as applicable, and, if approved, shall implement the improvements on the project.

The SDP is subject to reassessment and modification throughout the software life cycle. The SDP Revision Process is defined to provide structure for the technical manager to capture proposed revisions based on the project's status and stakeholder feedback.

Active communication among stakeholders is essential to the maintenance of an SDP. The technical manager is responsible for organizing face-to-face and remote meetings among technical and non-technical contributors to the project. Outputs from these meetings influence the SDP both in its initial development and its ongoing revisions.

9.4.3.1 Technical reviews

The technical manager shall schedule and conduct technical review meetings to assess and control technical performance (Table 1). The purpose of review meetings is to evaluate the work preformed and to determine if it satisfies the exit criteria of that phase and if progress to the next phase is feasible. Meetings may be conducted electronically or face-to-face.

Table 1 — Technical review meetings

Meeting Type	**Typical Frequency**	**Typical SDP Outcome**
Kick-off meeting	Once per project	Initial SDP may be approved.
Technical status meeting	Frequently, with regular intervals subject to the pace of project iterations and the need for communication	Revisions to the SDP may be proposed.
Review meeting	Occasionally, with intervals subject to progress against milestones, the need for a project decision and availability of stakeholders	Revisions to the SDP may be adopted.

a) The kick-off meeting. The kick-off meeting may be scheduled once per project, or once per iteration. The technical manager is responsible for this meeting.

Items discussed at this meeting reflect the SDP and shall include the following information from kick-off-meeting participants:

1) The acquirer makes project priorities known and distinguish project requirements from optional features.
2) The technical manager describes risks to completion and delays.
3) The technical manager describes the software life cycle model and methods.
4) The technical manager describes the team organization and the expected contributions of team members to the project.
5) The technical manager and acquirer agree on a timeline for the project and expected participation by the acquirer.

b) Technical status meetings. The technical status meeting brings together the technical manager, developers, designers, QA representatives and other technical professionals working on the project. The acquirer is often not present.

At the technical status meeting, participants share with one another technical information relevant to their progress on the project, including the following updates:

1) work completed since the previous technical status meeting;
2) work intended to be completed before the next technical status meeting; and
3) open questions requiring technical or managerial input from other participants.

Technical status meetings are held at regular intervals that are subject to the cycles of the project. For example, some Evolutionary methodologies call for daily technical status meetings, whereas some Once-through models call for weekly or monthly technical status meetings. The technical manager shall determine the periodic cycle of these meetings.

The technical manager collects and prioritizes proposed revisions to the SDP or project master schedule as a result of status meetings. For example, a developer may report that some piece of code is taking longer than expected, a delay that could impact the timeline of the overall project. As a result, the technical manager may propose updating the schedule with revised timelines, reducing the scope of the work, or bringing in additional developers to work on the project. In many organizations, the technical manager does not make substantial changes to the project alone, however, but proposes them to be considered by the project manager or other stakeholders. Therefore, these proposed revisions are considered provisional until they are approved using the project's change control procedures.

c) Review meetings. Review meetings are intended to document the work completed to-date to stakeholders who do not participate in regular technical status meetings. The technical team may demonstrate the deliverables for validation.

The audience for these demonstrations comprises several parties, each with a stake in the outcome of the project. The acquirer is often present at these meetings. Additional participants may include any or all of the project stakeholders. These participants may provide feedback to the functionality presented, which is collected by the technical manager and prioritized to address, ignore, or revise.

The technical manager reports the proposed revisions to the SDP that have been gathered from technical review meetings. The acquirer accepts the proposed revisions or collaborates with the technical manager to find alternative solutions to technical barriers.

The progress of the project is reviewed and its impact on the SDP is assessed. The technical manager should present the deliverables of the project to demonstrate their suitability, effectiveness, and success relative to expectations described in the SDP. During the early phases of the project, the technical manager may present non-software deliverables such as paper prototypes and design artifacts. During the software development phases, the technical manager may present early-stage software deliverables, such as a prototype user interface which will eventually be replaced.

Review meetings shall be held at intervals based on elapsed time or milestones accomplished. In some software development models, review meetings are held at the end of each iteration. In others, the review meeting may be held after a certain number of deliverables have been completed, depending on the progress of the project and availability of stakeholders.

9.5 Decision management

The technical manager shall record rationales that inform key decisions made throughout execution of the SDP. The meaning of "key decisions" and the approach for providing the rationale shall also be described in the SDP. Although the technical manager is not responsible for project management, he or she takes project management issues into consideration when making technical decisions.

Often, requests for decisions are based on technical issues. If a decision depends upon system states or modes, this dependency shall be indicated.

NOTE ISO/IEC/IEEE FDIS 12207:2017, 6.3.3 provides tasks and processes related to decision management.

Decision management contributes to the outcomes and activities of development planning and technical management. The following is useful to this effect:

a) The decision strategy shall help ensure that project stakeholders are involved in the decision management process, both to inform stakeholders of key decisions and to draw on stakeholders' experience and knowledge. The decision strategy should specify the roles and responsibilities of project stakeholders involved in the decision management process. The extent of involvement and agreement of stakeholders depends on the priority of the decision and should be stated in the decision strategy.

b) The project manager, technical manager, and other key stakeholders in the decision process shall monitor the implementation of each decision to confirm that problems have been effectively resolved, that any adverse trends have been reversed, and that the project has taken advantage of opportunities.

c) The project manager shall verify that the records of decisions implemented due both to problems and to opportunities are maintained in a manner that supports auditing and learning from experience.

d) The decision strategy should include identification of decision makers and authorities, decision categories and prioritization. Decision categories may include:

    1) implementation options for the requirements of the product;
    2) entry and exit criteria for life cycle stages;
    3) trade studies to objectively weigh options that are required to support a decision;
    4) make-or-buy decisions for components of the product or system to be delivered;
    5) issues affecting the organization's strategic business objectives;
    6) categories of risk for which formal mitigation plans are required; and
    7) trigger points for contingency plans.

e) Problems or opportunities arising during project execution should be recorded, categorized, and promptly and objectively reported, along with the possible corrective action or resolution.

f) The project manager, technical manager, and other key stakeholders in the decision process should identify the circumstances and need for a decision.

10 Information items: Technical plans

10.1 Software Development Plan related to other plans

The SDP specifies the technical management planning and control processes that are needed to achieve the software development objectives, while the project management plan specifies the project management processes that are applicable across the project. The SDP shall be consistent with the statement of project scope and planning for project initiation, project work, project acquisition and supply, project assessment and control, and project closeout of the project management plan.

Typically, a project has one or more project-specific engineering plans which define applicable policies and activities for the applicable system life cycle processes in the scope of the project. The technical manager uses the technical plans to guide and track software development activities. The project-specific engineering plan should be compliant with project management plans; organizational plans, capabilities, and constraints; and customer expectations.

Since each project may have unique life cycle dimensions, tolerance for risk, and need for data, the SDP should be adapted for each project, consistent with organizational policies and procedures.

The SDP shall be prepared and updated throughout the life of the project to guide and control the technical efforts of the project. The SDP (or engineering plan) should reflect an integrated technical management effort that addresses all factors associated with meeting the project's system life cycle requirements. If an evolutionary or incremental development strategy is to be pursued, the engineering plan should address the development strategy for initial product development and insertion of incremental capability or technology enhancements.

The SDP is the key project plan for managing the technical effort on a software development project. SDP development is tightly coupled with PMP development and content. The SDP may be part of the PMP, and the SDP may also be part of the SEMP. Depending on the complexity of the project, plans may be prepared for each life-cycle process, or combined into a single SDP or Software Engineering Plan.

NOTE ISO/IEC/IEEE 16326 specifies required content and format for project management plans covering software projects.

The technical manager shall place approved technical management plans under configuration management.

The SDP may include the content of the following plans referenced in ISO/IEC/IEEE FDIS 12207:2017, Annex B, or, if these plans are documented separately, they shall be referenced in the SDP.

1) Organizational project-enabling process plans:
   a) Process Improvement Plan
   b) Quality Management Plan
   c) Knowledge Management Plan

2) Technical Management process plans:
   a) Project plan (e.g., Project Technical Management Plan, Systems or Software Engineering Management Plan, Software Development Plan)
   b) Risk Management Plan
   c) Configuration Management Plan

3) Technical process plan:
   a) Operations continuity plan (for Development Operations)

10.2 Content of the Software Development Plan

10.2.1 Content Overview

The SDP presents how the organization or project plans to conduct development activities (the software implementation strategy).

NOTE ISO/IEC/IEEE 15289:2015, 10.21, defines the elements of an SDP. A sample SDP outline is shown in Annex B.

The following content shall be addressed by the SDP with description or reference to related materials.

1) Scope
   - 1.1) Identification
   - 1.2) System overview
   - 1.3) Document overview
   - 1.4) Relationship to other plans
2) Referenced documents
3) Overview of required work
4) Project organization and control
5) Plans for performing detailed software development activities
   - 5.1) Establishing a software development environment
   - 5.2) Software system requirements definition
   - 5.3) Architecture and design activities
   - 5.4) Software implementation (including construction)
   - 5.5) Software system integration
   - 5.6) Software system verification and validation
   - 5.7) Software transition and release management
   - 5.8). Software sustainment
   - 5.9) Software defect and problem management
   - 5.10) Software retirement
6) Schedule and budget summary

10.2.2 Detailed content of the SDP

10.2.2.1 Scope

Identify the purpose, scope, and objectives of the project and the products to be delivered in terms of the software or technical effort. This clause should also describe significant considerations of scope or objectives that are to be excluded from the project or the resulting software development product.

A reference to the approved statement of product requirements shall be provided.

Do not replicate information found in the Project Management Plan beyond that necessary to provide a brief overview of the project. Reference the Project Management Plan for more detailed information on the project description, purpose, scope, and objectives.

10.2.2.2 Referenced documents

Identify the objectives and standards to be used in the software development process. List internal documents that influence the design decisions and implementation choices of the project.

10.2.2.3 Overview of required work

Describe the required work of the project, including assumptions on which the software development effort is based and identify imposed constraints on project factors such as the schedule, budget, resources, software to be reused, acquirer software to be incorporated, technology to be employed, and product interfaces to other products.

Also provide a brief statement of the business or system needs to be satisfied and the expected life span of the software system. Include a concise summary of the project software development objectives, the products to be delivered to satisfy those objectives, and the methods by which satisfaction is determined. The project statement of purpose shall describe the relationship of this project to other projects, and, as appropriate, how this project is integrated with other projects or ongoing work processes.

10.2.2.4 Project organization and control

1) Technical organization. Provide a summary description of the project's technical organization and the team structure and describe the software development team, their roles, responsibilities and authorities, and how the team relates to internal and external entities. The SDP may refer to the Project Management Plan for detailed project organization information.

NOTE 1 The Project Management Plan is typically the primary document for project organization information and identifies interfaces to organizational entities external to the project, describes the project’s internal organizational structure, and defines roles and responsibilities for the project.

2) Project assessment and control. Describe the assessment and control activities that are applied to the software development effort. Specify the procedures necessary to assess and control the product requirements; the technical project scope, schedule, budget, and resources; the quality and timeliness of acquired products from subcontractors; and the quality of work processes and work products. Some additional concerns highlighted for technical control include design capture, interface management, data management, coordinated engineering scheduling, establishment of measures including Technical Performance Measures (TPMs), technical reviews and identification, and transitioning of critical technologies. Explain how periodic monitoring of measurements and key milestones indicate when management action is needed. The elements of the technical control planning should be consistent with the project's standards, policies, and procedures for project control as well as with any contractual agreements for project control.

NOTE 2 Separate plans, e.g., requirements management plan, measurement plan, and quality assurance plan, are commonly developed to support control. The SDP may reference these supplemental plans.

10.2.2.5 Plans for performing detailed software development activities

1) Software development environment. Identify the system or software life-cycle model to be used to satisfy the product or service requirements, based on the project’s scope, magnitude and complexity. Map the development process and activities to the selected life-cycle model. Identify the strategy, methods, and tools to be used in development and test.

Identify the locations where work will be performed and the distinct environments to be established for development, testing, and other purposes.

2) Software system requirements definition. Identify how requirements will be identified, traced, and managed.

3) Architecture and design activities. Briefly describe the methods for defining, modeling, and describing the software system architecture and design. Explain how the design is traceable to the requirements and the implemented software.

4) Software implementation (including construction). Describe how the various inputs into the software development effort will be implemented. Identify systems, notations and naming conventions used in development for configuration management.

5) Software system integration. Describe how software elements (units, components or items) will be integrated in a coordinated development effort to meet cost, schedule, and performance objectives for the software system. Provide a brief description of the approach and methods planned to assure integration of the engineering specialties needed to meet project objectives.

6) Software system verification and validation (testing). Explain how requirements, including non-technical requirements such as safety and security, will be verified and validated for the software system.

7) Software integration into hardware. Describe how software elements will be integrated with relevant hardware elements to realize a comprehensive system.

8) Software transition and release management. Describe the activities for control of software versions, software distribution, and transition of software between environments.

9) Software sustainment. Describe the approach for improving the software and correcting defects once the software is released.

10) Software defect and problem management. Explain how software defects and technical problems will be identified, recorded, and resolved (corrective action). Describe activities for quality assurance and process improvement.

10.2.2.6 Schedule and budget summary

Provide a brief summary of the schedule and budget for the project. Budget may be presented in terms of effort or cost. At a high level, identify the major work activities and supporting processes as, for example, those depicted by the top levels of the Work Breakdown Structure. The schedule shall reflect the project duration with initiation of major activities, availability of draft and final deliverables, and key milestones (decision gates) for the software development effort. Include the schedule for the next iteration. A detailed schedule is often maintained as a separate information item, and referenced in the SDP. For evolutionary and incremental life cycle models, the schedule may include a high level overview schedule, with details only for the first phase or increment. Identify those activities that impose the greatest time constraints on the project.

Annex A
(informative)

Generic content for a plan

Purpose: Define when, how, and by whom specific processes or activities are to be performed.
A plan shall include the following elements:

a) date of issue and status;
b) scope;
c) issuing organization;
d) references (applicable policies, laws, standards, contracts, requirements, and other plans and procedures);
e) approval authority;
f) approach for technical and management review and reporting;
g) other plans (plans or task descriptions that expand on the details of a plan);
h) planned activities and tasks;
i) identification of tools, methods, and techniques;
j) schedules;
k) budgets and cost estimates;
l) resources and their allocation;
m) responsibilities and authority, including the senior responsible owner and immediate process owner;
n) interfaces among parties involved;
o) risks and risk identification, assessment and mitigation activities;
p) quality assurance and control measures;
q) environment, infrastructure, security, and safety;
r) training;
s) glossary;
t) change procedures and history; and
u) termination process.

Annex B
(informative)

Sample Software Development Plan outline

The following is an example of an SDP outline. It is not required to treat every topic required in this document in the same order, under the same wording for its title, or with the same level of detail, depending on the nature of the software, implementation methods, life cycle model, and scope of the project.

1 Scope
- 1.1 Identification
- 1.2 System overview
- 1.3 Document overview
- 1.4 Relationship to other plans

2 Referenced documents

3 Overview of required work

4 Project organization and resources
- 4.1 Project organization
- 4.2 Project resources
- 4.2.1 Software and system resources
- 4.2.2 Human resources
- 4.2.3 Infrastructure
- 4.2.4 Funding
- 4.3 Project control
- 4.3.1 Project planning and oversight
    - 4.3.1.1 Software development planning (covering updates to this plan)
    - 4.3.1.2 CI test planning
    - 4.3.1.3 System test planning
    - 4.3.1.4 Software installation planning
    - 4.3.1.5 Software transition planning
    - 4.3.1.6 Following and updating plans, including the intervals for management review
    - 4.3.1.7 Joint technical and management reviews
        - 4.3.1.7.1 Joint technical reviews, including a proposed set of reviews
        - 4.3.1.7.2 Joint management reviews, including a proposed set of reviews
    - 4.3.1.8 Project estimation
- 4.3.2 Other software development activities
    - 4.3.2.1 Risk management, including known risks and corresponding strategies
    - 4.3.2.2 Software measurement and management indicators
    - 4.2.3.3 Safety management
    - 4.3.2.4 Security and privacy
    - 4.3.2.5 Subcontractor management
    - 4.3.2.6 Interface with software independent verification and validation (IV&V) agents
    - 4.3.2.7 Coordination with developers
    - 4.3.2.8 Improvement of Technical Management processes

4.3.2.9 Improvement of Technical processes

5 Plans for performing general software development activities

5.1 Software development model

5.2 General plans for software development

5.2.1 Software development methods

5.2.2 Standards for software products

5.2.3 Reusable software products

5.2.3.1 Incorporating reusable and COTS software

5.2.3.2 Developing reusable software products

5.2.4 Handling of   requirements

5.2.4.1 Safety assurance

5.2.4.2 Security assurance

5.2.4.3 Privacy assurance

5.2.4.4 Assurance of other critical quality characteristics

5.2.5 Computer hardware resource utilization

5.2.6 Recording rationale

5.2.7 Access for acquirer review

5.2.8 Software development tools

6 Plans for performing detailed software development activities

6.1 Involvement of user representatives in software development

6.2 Establishing a software development environment

6.3 Software System requirements analysis

6.3.1. Stakeholder requirements

6.3.2 Analysis of user input

6.3.3 Operational concept

6.3.4 System requirements

6.3.5 Software requirements analysis

6.4 Software System and architectural design

6.4.1 System-wide design decisions

6.4.2 System architectural design

6.4.3 CI-wide design decisions

6.4.4 CI architectural design

6.4.5 CI detailed design

6.5 Software integration into hardware

6.6 Software implementation and unit testing

6.6.1 Software implementation

6.6.2 Preparing for unit testing

6.6.3 Performing unit testing

6.6.4 Revision and retesting

6.6.5 Analyzing and recording unit test results

6.7 Unit integration and testing

6.7.1 Preparing for unit integration and testing

6.7.2 Performing unit integration and testing

6.7.3 Revision and retesting

6.7.4 Analyzing and recording unit integration and test results