INTERNATIONAL STANDARD

ISO/ IEC/IEEE 32675

First edition
2022-08

Information technology — DevOps — Building reliable and secure systems including application build, package and deployment

Technologies de l'information — DevOps — Création de systèmes fiables et sûrs notamment en matière de compilation, paquetage et déploiement d'applications

Reference number
ISO/IEC/IEEE 32675:2022(E)

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

Email: stds.ipr@ieee.org
Website: www.ieee.org

Published in Switzerland

Foreword

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 non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO/IEC documents should be noted (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

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.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see https://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

ISO/IEC/IEEE 32675 was prepared by the Systems and Software Engineering Standards Committee of the IEEE Computer Society (as IEEE Std 2675-2021) and drafted in accordance with its editorial rules. It was adopted, under the "fast-track procedure" defined in the Partner Standards Development Organization cooperation agreement between ISO and IEEE, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and systems engineering*.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment

Developed by the

Software & Systems Engineering Standards Committee
of the
IEEE Computer Society

Approved 9 February 2021

IEEE SA Standards Board

Abstract: Technical principles and processes to build, package, and deploy systems and applications in a reliable and secure way are specified. Establishing effective compliance and information technology (IT) controls is the focus. DevOps principles presented include mission first, customer focus, left-shift, continuous everything, and systems thinking. How stakeholders, including developers and operations staff, can collaborate and communicate effectively is described. The process outcomes and activities herein are aligned with the process model specified in ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2015.

Keywords: agile, continuous delivery, continuous deployment, continuous integration, DevOps, IEEE 2675™, left-shift

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

PDF: ISBN 978-1-5044-7407-8 STD24616
Print: ISBN 978-1-5044-7408-5 STDPD24616

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE Standards documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page (https://standards.ieee.org/ipr/disclaimers.html), appear in all standards and may be found under the heading "Important Notices and Disclaimers Concerning IEEE Standards Documents."

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE SA) Standards Board. IEEE develops its standards through an accredited consensus development process, which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE Standards are documents developed by volunteers with scientific, academic, and industry-based expertise in technical working groups. Volunteers are not necessarily members of IEEE or IEEE SA, and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE does not warrant or represent the accuracy or completeness of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to results and workmanlike effort. In addition, IEEE does not warrant or represent that the use of the material contained in its standards is free from patent infringement. IEEE Standards documents are supplied "AS IS" and "WITH ALL FAULTS."

Use of an IEEE standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity, nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: THE NEED TO PROCURE SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE is the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, nor be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that the presenter's views should be considered the personal views of that individual rather than the formal position of IEEE, IEEE SA, the Standards Committee, or the Working Group.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE or IEEE SA. However, **IEEE does not provide interpretations, consulting information, or advice pertaining to IEEE Standards documents**.

Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its Societies and Standards Coordinating Committees are not able to provide an instant response to comments, or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in evaluating comments or in revisions to an IEEE standard is welcome to join the relevant IEEE working group. You can indicate interest in a working group using the Interests tab in the Manage Profile & Interests area of the IEEE SA myProject system. An IEEE Account is needed to access the application.

Comments on standards should be submitted using the Contact Us form.

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not constitute compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Data privacy

Users of IEEE Standards documents should evaluate the standards for considerations of data privacy and data ownership in the context of assessing and using the standards in compliance with applicable laws and regulations.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under US and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate licensing fees, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400; https://www.copyright.com/. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every 10 years. When a document is more than 10 years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit IEEE Xplore or contact IEEE. For more information about the IEEE SA or IEEE's standards development process, visit the IEEE SA Website.

Errata

Errata, if any, for all IEEE standards can be accessed on the IEEE SA Website. Search for standard number and year of approval to access the web page of the published standard. Errata links are located under the Additional Resources Details section. Errata are also available in IEEE Xplore. Users are encouraged to periodically check for errata.

Patents

IEEE Standards are developed in compliance with the IEEE SA Patent Policy.

IMPORTANT NOTICE

IEEE Standards do not guarantee or ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. IEEE Standards development activities consider research and information presented to the standards development group in developing any safety recommendations. Other information about safety practices, changes in technology or technology implementation, or impact by peripheral systems also may be pertinent to safety considerations during implementation of the standard. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

Participants

At the time this IEEE standard was completed, the DevOps Working Group had the following membership:

Bob Aiello, *Chair*
Lynn Robert Carter, *Secretary*

Devora Aiello
Sarah Baker
Jaynee Beach
Kristian Beckers
Subroto Bhattacharya
Paul Bruce
Simon Goldsmith
Victoria Hailey
Bob Jenkins
Daniel Katzman
Ruth Lennon
Nithyanandam Mathiyazhagan
Malu Milan
Harvey Nusz
Martin Radley
Tafline Ramos
Annette Reilly
Ayhan Tek
Mark Underwood
Altaz Valani
Chris Walker
Robert J. White
Steve Woodward
Hasan Yasar

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

Bob Aiello
Johann Amsenga
Bakul Banerjee
William Bearden
Juris Borzovs
Pieter Botman
Lynn Robert Carter
Manuel Castro
Ronald Dean
Teresa Doran
Andrew Fieldsend
Dan Friedman
David Fuschi
Louis Gullo
Jon Hagar
Victoria Hailey
Werner Hoelzl
Piotr Karocki
Daniel Katzman
Ralph Kearfott
Stuart Kerry
Edmund Kienast
Yongbum Kim
Naga Sai Kruthiventi
Thomas Kurihara
Jim Lewis
Johnny Marques
Rajesh Murthy
Nick S. A. Nikjoo
Joanna Olszewska
Beth Pumo
R. K. Rannow
Annette Reilly
Robert Schaaf
Stefan Schlichting
Stephen Schwarm
Subrato Sensharma
Carl Singer
Friedrich Stallinger
Thomas Starai
Walter Struppler
Gerald Stueve
Max Turner
Mark-Rene Uchida
Altaz Valani
John Vergis
Marcel Winandy
Hasan Yasar
Yu Yuan
Oren Yuen
Janusz Zalewski

When the IEEE SA Standards Board approved this standard on 9 February 2021, it had the following membership:

Gary Hoffman, *Chair*
Vacant Position, *Vice Chair*
John D. Kulick, *Past Chair*
Konstantinos Karachalios, *Secretary*

Edward A. Addy
Doug Edwards
Ramy Ahmed Fathy
J. Travis Griffith
Thomas Koshy
Joseph L. Koepfinger*
David J. Law
Howard Li
Daozhuang Lin
Kevin Lu
Daleep C. Mohla
Chenhui Niu
Damir Novosel
Annette Reilly
Jon Walter Rosdahl
Dorothy Stanley
Mehmet Ulema
Lei Wang
F. Keith Waters
Karl Weber
Sha Wei
Howard Wolfman
Daidi Zhong

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 2675™-2021, IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment.

The complexity of software systems has increased to an unprecedented level. This has led to new opportunities, but also to increased challenges for the organizations that create and utilize systems. One of the greatest challenges has been to address the increased rate of change in modern development methodologies, including agile and even rapid iterative development. These challenges exist throughout the life cycle of a system and at all levels of architectural detail. This document highlights the manner in which DevOps can help address the challenges inherent in accelerated development methodologies and achieve end user goals for increased productivity and quality.

DevOps is an interdisciplinary approach and means to enable the realization of successful software systems. It focuses on defining stakeholder needs and required functionality early in the development cycle, documenting requirements, and performing design synthesis and system validation while considering the complete problem. It integrates the disciplines and specialty groups into a team effort forming a structured development process that proceeds from concept to production to operation and maintenance. It considers both the business and the technical needs of stakeholders with the goal of providing a quality product that meets the needs of users and other applicable stakeholders. This life cycle spans the conception of ideas through to the retirement of a system. It provides the processes for acquiring and supplying systems. It helps improve communication and cooperation among the parties that create, utilize, and manage modern software systems. In addition, this framework provides for the assessment and improvement of the life cycle processes.

This document is appropriate both for organizations that are unused to applying engineering process standards, and for those who have used longstanding standards, who have the goal of implementing effective information technology (IT) controls, embracing and managing risk, while enabling more rapid development (higher velocity). Organizations that are already embracing IEEE standards can find IEEE Std 2675 to be essential in helping them to implement the DevOps transformation. Organizations that choose IEEE Std 2675 as their first industry standard can subsequently apply a broader family of IEEE standards.

This document is closely aligned with the life cycle processes in ISO/IEC/IEEE 12207:2017 [B14][1] and ISO/IEC/IEEE 15288:2015. Configuration management is the basis of DevOps and hence it is also closely aligned with IEEE Std 828™ [B3], along with other related standards.

[1] The numbers in brackets correspond to those of the bibliography in Annex A.

Contents

IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment

1. Overview

1.1 Scope

This document provides requirements and guidance on the implementation of DevOps to define, control, and improve software life cycle processes. It applies within an organization or a project to build, package, and deploy software and systems in a secure and reliable way. This document specifies practices to collaborate and communicate effectively in groups including development, operations, and other key stakeholders.

This document applies a common framework for software life cycle processes, with well-defined terminology. It contains processes, activities, and tasks that are to be applied to the full life cycle of software systems, products, and services, including conception, development, production, utilization, support, and retirement. It also applies to the acquisition and supply of software systems, whether performed internally or externally to an organization. These life cycle processes are accomplished through the involvement of stakeholders, with the ultimate goal of achieving customer satisfaction. The life cycle processes of this document can be applied concurrently, iteratively, and recursively to a software system and incrementally to its elements.

This document applies to software systems, products, and services, and the software portion of any system. Software includes the software portion of firmware. Those aspects of system definition needed to provide the context for software systems, products, and services are included.

There is a wide variety of software systems in terms of their purpose, domain of application, complexity, size, novelty, adaptability, quantities, locations, life spans, and evolution. This document describes the processes that comprise the life cycle of software systems. It therefore applies to one-of-a-kind software systems, software systems for wide commercial or public distribution, and customized, adaptable software systems. It also applies to a complete stand-alone software system and to software systems that are embedded and integrated into larger, more complex, and complete systems.

1.2 Purpose

The purpose of this standard is to specify required practices for operations, development, and other key stakeholders to collaborate and communicate to deploy systems and applications in a secure and reliable way. This document provides a defined set of processes and methods to facilitate DevOps principles and practices, including improved communication between stakeholders throughout the systems life cycle, not just during development and operations. This document is written for DevOps stakeholders, which includes, but is not limited to, acquirers, suppliers, developers, integrators, operators, maintainers, managers, quality assurance managers, compliance managers, auditors, and users of software systems, products, and services. It can be used by a single organization in a self-imposed mode or in a multi-party situation. Parties can be from the same organization or from different organizations, and the situation can range from an informal agreement to a formal contract.

The processes in this document can be used as a basis for implementing DevOps while establishing organizational environments, e.g., methods, procedures, techniques, tools, and trained personnel. The processes in this document provide guidance on the use of DevOps principles and practices for processes used by an organization to construct software life cycle models appropriate to its products and services. An organization, depending on its purpose, can select and apply an appropriate subset to fulfill that purpose.

This document can be used in one or more of the following modes:

a) By an organization—to establish DevOps principles and practices in support of an environment of desired processes. These processes can be supported by an infrastructure of methods, procedures, techniques, tools, and trained personnel. The organization may then employ this environment to perform and manage its projects and progress software systems through their life cycle stages. In this mode, this document is used to assess conformance of a declared, established environment to its provisions.

b) By a project—to establish DevOps principles and practices to help select, structure, and employ the elements of an established environment to provide products and services. In this mode, this document is used in the assessment of conformance of the project to the declared and established environment.

c) By an acquirer and a supplier—to establish DevOps principles and practices to help develop an agreement concerning processes and activities. Via the agreement, the processes and activities in this document are selected, negotiated, agreed to, and performed. The acquirer and supplier can be part of the same organization or separate organizations.

d) By process assessors—to establish DevOps principles and practices in a process reference model for use in the performance of process assessments that may be used to support organizational process improvement.

1.3 Word usage

The word *shall* indicates mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (*shall* equals is *required to*).[2, 3]

The word *should* indicates that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required (*should* equals is *recommended that*).

[2] The use of the word *must* is deprecated and cannot be used when stating mandatory requirements, *must* is used only to describe unavoidable situations.
[3] The use of *will* is deprecated and cannot be used when stating mandatory requirements, *will* is only used in statements of fact.

The word *may* is used to indicate a course of action permissible within the limits of the standard (*may* equals is *permitted to*).

The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals is *able to*).

2. Normative references

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

This document has no normative references.

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary Online* should be consulted for terms not defined in this clause.[4]

For additional terms and definitions in the field of systems and software engineering, see ISO/IEC/IEEE 24765 [B20], which is published periodically as a "snapshot" of the SEVOCAB (Systems and software Engineering Vocabulary) database and is publicly accessible at computer.org/sevocab.

NOTE—While the aim is to provide consistency in terminology throughout the IEEE standards, it is worth noting that, particularly from the DevOps perspective, there are often alternative terms for similar roles or processes. The applicability of terms to development, operations, testing, security, and performance was separately considered so that the terminology used was applicable in every case.[5]

aligned: Group agreement and alliance to one or more shared objectives.

NOTE—Key concepts are that each member understands critical inputs (i.e., information, context, and constraints), acts according to a plan that is communicated to all members, accepts responsibility for their part in requisite activities and tasks, and harmoniously collaborates with other members and external resources.

archive: Location of system elements that are no longer present in runtime environments, but are available for examination for audit, regulatory, and other processes.

audit: Independent, continuous examination of a work product or set of work products to assess compliance with specifications, standards, contractual agreements, or other criteria for the purpose of providing assurance against risk.

NOTE 1—Generating evidence of information technology (IT) controls that support audit is often automated where practical.

[4] *IEEE Standards Dictionary Online* is available at: http://dictionary.ieee.org. An IEEE Account is required for access to the dictionary, and one can be created at no charge on the dictionary sign-in page.
[5] Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

NOTE 2—Audit can be orchestrated and integrated as part of a DevOps pipeline.

build: Process of generating an executable and testable system from source versions or baselines. (IEEE Std 828™.)

business value: Strategic priorities set forth by the business as it relates to revenue, cost, risk, security, privacy, ethics, and compliance.

competency: Ability to demonstrate and apply the combination of knowledge, formal and informal skills, training, experience, and behavioral attributes to achieve intended organizational and technical results.

compliance: Continual fulfillment to internal and external requirements, rules, and regulations. (SEVOCAB.)

configuration management: Technical and organizational activities, comprising configuration identification, control, status accounting, and auditing.

continuous delivery: Software engineering practices that allow for frequent releases of new systems (including software) to staging or various test environments through the use of automated tools.

continuous deployment: Automated process of deploying changes to production by verifying intended features and validations to reduce risk.

continuous integration: Technique that continually merges artifacts, including source code updates from all developers on a team, into a shared mainline to build and test the developed system.

continuous risk management: Continuous process, which may be automated, that identifies, applies, and monitors controls to treat risks for a planned activity, project, or program, to achieve a desired outcome.

cryptographic hash: Method to verify the authenticity of a system element or software via the production of a checksum.

data-driven: Informing an activity by evidence.

defect: Imperfection or deficiency in a work product or characteristic that does not meet its requirements or specifications.

deployment: Stage of a life cycle in which a system is put into operation and transition issues are resolved.

DevOps: Set of principles and practices which enable better communication and collaboration between relevant stakeholders for the purpose of specifying, developing, and operating software and systems products and services, and continuous improvements in all aspects of the life cycle.

disposal: Removal or archiving, but not deletion of an artifact, so it can be made available for traceability and auditability.

error: Discrepancy between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition. (ISO/IEC/IEEE 15026-1:2019 [B15], 3.4.5.)

infrastructure: Facilities such as power, cooling, and physical security of the data center, networking, hardware, and software needed to support the systems life cycle and maintain information technology (IT) services.

NOTE—Does not include the associated people, processes, or documentation. In DevOps, software-defined infrastructure enables elasticity.

infrastructure as code (IaC): Definition, management, and provision of infrastructure components using software.

NOTE—In DevOps, infrastructure as code facilitates the automation of the systems life cycle enabling consistency, performance, and security across the system and resources.

knowledge management: Multi-disciplinary process of obtaining, preserving, sharing, using, and refreshing knowledge.

NOTE—In DevOps, knowledge management guides and facilitates the automation of system operations, system problem identification and remediation, and reporting system health.

left-shift: Prioritizing the involvement of relevant stakeholders in applying quality activities, security, privacy, performance, verification, and validation earlier in the life cycle.

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

NOTE—In DevOps, the systems life cycle is supported by automated elements that produce meaningful and actionable audit logs.

life cycle model: Framework of processes and activities concerned with the life cycle, which can be organized into stages, acting as a common reference for communication and understanding.

machine readable: Pertaining to data in a form that can be automatically generated by and input to a computer.

minimum viable product: Version of a work product with just enough features and requirements to satisfy early customers and provide feedback for future development.

monitoring: Determining the status of a system, a process, or an activity. (ISO/IEC 19770-1:2017, 3.35.)

package: To combine related components into a single, deployable item.

platform as a service (PaaS): Provision of a complete environment of IT resources, such as programming languages, libraries, services, and tools supported by the service provider.

NOTE—The level of control over the service provided to the customer can vary with the service provider. In DevOps, PaaS is automated as part of the DevOps pipeline.

pipeline: Software or hardware design technique in which the output of one process serves as input to a second, the output of the second process serves as input to a third, and so on, often with simultaneity within a single cycle time.

policy: Intentions and direction of an organization, as formally expressed by its top management. (ISO/IEC 19770-1:2017, 3.43.)

NOTE—Direction includes mandates set by the organization.

portfolio: Projects, programs, and operations managed as a group to achieve business objectives.

portfolio management: Centralized management of one or more portfolios to achieve business objectives.

problem: Difficulty or uncertainty experienced by one or more persons, resulting from an unsatisfactory encounter with a system in use.

quality assurance: Part of quality management focused on continually providing confidence that requirements are being fulfilled.

role-playing session: Technique of engagement including the relevant stakeholders in an interactive simulation of the dynamic interactions that are part of the system design.

software as a service (SaaS): Access to resources from client devices through thin client interface or program interface.

NOTE—The level of control over the resource provided to the consumer can vary with the service provider. In DevOps, SaaS can be automated as part of the DevOps pipeline.

software life cycle: Project-specific sequence of activities that is created by mapping the activities of a standard onto a selected software life cycle model. (SEVOCAB.)

stage: Period within the life cycle of an entity that relates to the state of its description or realization. (SEVOCAB.)

stakeholder: Individual, group, or organization having a right, share, claim, or interest in a system or in its possession of characteristics that meet their needs and expectations. For example, legal, financial, risk, compliance, security, privacy, and end user organizations; supporters, developers, producers, trainers, implementers, maintainers, operations, disposers, acquirers, supplier organizations, and regulatory bodies.

NOTE—Some stakeholders can have interests that oppose each other or oppose the system.

validation: Confirmation in a timely manner, through automated techniques where possible, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled.

NOTE—A system is able to accomplish its intended use, goals, and objectives (i.e., meet stakeholder requirements) in the intended operational environment. The right system is operating to meet business objectives.

velocity: The rate of current work unit completion, measured as work units completed per fixed time period, such as story points, delivered features, functions, function points, user stories, use cases, or requirements completed in a given time period.

NOTE—Used as a measure of burndown rate or burnup rate.

verification: Confirmation in a timely manner, using automated techniques where possible, through the provision of objective evidence, that specified requirements have been fulfilled. (ISO 9000:2015.)

NOTE—Verification is a set of activities that compares a system or system element against the required characteristics. This includes, but is not limited to, specified requirements, design, descriptions, and the system itself. The system is operating as per business objectives.

3.2 Acronyms and abbreviations

ALM application life cycle management

CD continuous delivery, continuous deployment

CI configuration item, continuous integration

CM configuration management

COTS	commercial-off-the-shelf
DRP	disaster recovery plan
ETL	extract, transform, load
FCA	functional configuration audit
IaaS	infrastructure as a service
IaC	infrastructure as code
KPI	key performance indicator
MTTD	mean time to detection
MTTR	mean time to recovery
OLA	operations level agreement
OSS	open source software
QA	quality assurance
QM	quality management
QoS	quality of service
PaaS	platform as a service
SaaS	software as a service
SHA	secure hash algorithm
SLA	service level agreement
SLI	service level indicator
SLO	service level objective
SME	subject matter expert
SOI	system-of-interest
SOP	standard operating procedure
SoS	system of systems
TDD	test-driven development
V&V	validation and verification

4. Conformance

4.1 Compliance criteria

This document provides requirements for a number of processes suitable for usage during the life cycle of a software service, system, or product. Requirements stated for an organization may be applied to any size of organization, program, project, or entity.

Particular projects or organizations may not need to use all of the processes provided by this document. Therefore, implementation of this document typically involves selecting and declaring a set of processes suitable to the organization or project. Each process has a set of objectives (phrased as "outcomes") and a set of activities and tasks that represent one way to achieve the objectives.

Options for conformance are provided for needed flexibility in the application of this document. There are two criteria for claiming full conformance. Achieving either criterion suffices for conformance, and the chosen criterion (or criteria) shall be stated in the claim.

— Claiming "full conformance to outcomes" asserts that all of the required outcomes of the declared set of processes are achieved.

— Alternatively, claiming "full conformance to tasks" asserts that all of the requirements of the activities and tasks of the declared set of processes are achieved.

Users who implement the activities and tasks of the declared set of processes can assert full conformance to tasks of the selected processes. Some users, however, might have innovative process variants that achieve the objectives (i.e., the outcomes) of the declared set of processes without implementing all of the activities and tasks. These users can assert full conformance to the outcomes of the declared set of processes. The two criteria—conformance to task and conformance to outcome—are necessarily not equivalent since specific performance of activities and tasks can require, in some cases, a higher level of capability than just the achievement of outcomes.

NOTE 1—When this document is used to help develop an agreement between an acquirer and a supplier, clauses of this document can be selected for incorporation in the agreement with or without modification. In this case, it is more appropriate for the acquirer and supplier to claim compliance with the agreement than conformance with this document.

NOTE 2—An organization (for example, nation, industrial association, company) imposing this document as a condition of trade, can specify and make public the minimum set of required processes, outcomes, activities, and tasks which constitute suppliers' compliance with the conditions of trade.

Requirements of this document are marked by the use of the verb *shall*. Recommendations are marked by the use of the verb *should*. Permissions are marked by the use of the verb *may*. However, despite the verb that is used, the requirements for conformance are selected as described previously.

Demonstrable evidence may be represented as, for example, documented standard operating procedures (SOPs) or work practices, workflows showing process checks, quality metrics, or in-process review steps that are formally documented as part of the configuration management process. The specificity of the demonstrable evidence will vary with the business domain, organizational policies, and other factors. In DevOps, demonstrable evidence may be collected, where possible in real time and in an automated fashion, to verify conformance to a requirement.

4.2 Full conformance to outcomes

A claim of full conformance declares the set of processes for which conformance is claimed. Full conformance to outcomes is achieved by demonstrating that all of the outcomes of the declared set of processes have been achieved. In this situation, the provisions for activities and tasks of the declared set of processes are guidance rather than requirements, regardless of the verb form that is used in the provision. Full conformance to outcomes permits greater freedom in the implementation of conforming processes and can be useful for implementing processes to be used in the context of an innovative life cycle model.

One intended use of this document is to facilitate process assessment and improvement. For this purpose, the objectives of each process are written in the form of "outcomes" compatible with the provisions of ISO/IEC/IEEE 24774 [B21].

NOTE—ISO/IEC 33002 [B12] provides for the assessment of processes, which can be a basis for process improvement. Users intending process assessment and improvement can use the process outcomes written in this document as the process reference model required by ISO/IEC 33002.

4.3 Full conformance to tasks

A claim of full conformance declares the set of processes for which conformance is claimed. Full conformance to tasks is achieved by demonstrating that all of the requirements of the activities and tasks of the declared set of processes have been achieved. In this situation, the provisions for the outcomes of the declared set of processes are guidance rather than requirements, regardless of the verb form that is used in the provision.

NOTE—A claim of full conformance to tasks can be appropriate in contractual situations where an acquirer or a regulator requires detailed understanding of the suppliers' processes.

4.4 Tailored conformance

When this document is used as a basis for establishing a set of processes that do not qualify for full conformance, the clauses of this document are selected or modified. The tailored text, for which tailored conformance is claimed, is declared. Tailored conformance is achieved by demonstrating that the outcomes, activities, and tasks, as tailored, have been achieved. Small to medium enterprises can use tailored conformance as a technique to begin the process toward obtaining full conformance.

NOTE—Annex A of ISO/IEC/IEEE 12207:2017 [B14] provides requirements and guidance for tailoring a process.

5. DevOps concepts

5.1 Value of DevOps

This document is intended to demonstrate how DevOps is appropriate for building, packaging, and deploying reliable and secure systems. This document presents a more holistic view of DevOps than its literal meaning of combining development and operations resources and procedures to perform continuous integration, delivery, and deployment.

The term *DevOps* evolved from the availability of fully automated application build, package, and deployment tools, along with the recognition that information technology (IT) organizations were not

prepared to use those tools effectively. Developers and operations professionals traditionally had different (and, at times, conflicting) perspectives. An extreme, oversimplified characterization is that developers want to go fast and are mostly concerned with velocity (delivering value to end users as quickly as possible), while the operations team is concerned with assuring reliability and enforcing cybersecurity. In that sense, DevOps can appear to ignore the traditional safeguards of standard life cycle processes, such as separation of duties, multiple levels of reviews, and independently tested and infrequently scheduled releases. From that perspective, DevOps can appear to be unsuitable for organizations concerned with security and risk to critical applications.

In practice, DevOps aims to satisfy a dynamic and competitive marketplace that favors products that balance the V requirements (volume, velocity, variety, veracity, value, and others). DevOps seeks to achieve a balance between velocity and system reliability and stability. DevOps was created to provide solutions to constantly changing complex problems, where reducing organizational risk and improving security and reliability are critical requirements. The diverse international domains where reliable and secure systems are used often have rigorous regulatory and legal requirements. Therefore, DevOps requires identifying, engaging, and dynamically collaborating with a set of supplier and acquirer (producer and consumer) entities forming a holistic "ecosystem" of both internal and external organizations and stakeholders (e.g., customers, business units, support organizations, suppliers, collaborators, and partners).

This clause presents key concepts of DevOps for reliable and secure systems: its main principles and practices, its changes to organizational culture, and its relationship to traditional life cycle processes.

5.2 DevOps principles

5.2.1 Business or mission first

DevOps focuses on business and organizational goals ahead of procedural and technical considerations. DevOps utilizes information-rich feedback loops to understand progress and threats to attaining business and mission goals. Taking a business or mission first view helps to balance the concerns of risk and the activities which provide the most value to the customer. Continuously finding a dynamic balance between opportunities and risks and applying risk assessment and treatment at the organizational level enables the realization of DevOps without thrashing and wasted resources. Realizing the full promise of DevOps requires maintaining a strategic balance between honoring today's commitments while enabling the organization to survive risks and develop the capabilities to move forward.

5.2.2 Customer focus

DevOps takes a customer-centric view, prioritizing and designing work to deliver value to the customer, as well as identifying and managing risk. In short, if it makes sense for the customer and meets a customer need, then it is likely to be the right approach from a DevOps perspective. For example, for both customers and suppliers, privacy is a customer focus: not only protection of individuals' data, but also protection of enterprise data. Privacy may include data categorization (e.g., health data, financial data) and classification-driven methods (e.g., confidential, restricted, internal use, public) as often required by regulatory and legal requirements.

DevOps relies on keeping stakeholders informed and aware of changes that can impact them, by means of automation when practicable. To reduce information overload and to allow stakeholders to focus on those changes that exceed their individual risk thresholds, typical tools include automated support to inform involved stakeholders when an issue is approaching their individual threshold and when actions are initiated if a threshold is crossed. Especially when systems have multiple stakeholders, collaboration mechanisms to agree on decisions and resolve conflicting concerns in real time should be designed and

established before issues occur. Figuring out what to do during a crisis is much more difficult when those involved are overloaded and under significant stress.

Example: An existential financial risk to one stakeholder can be a life-critical risk to another.

5.2.3 Left-shift and continuous everything

The normal DevOps practice is information-driven, risk-based, continuous everything. DevOps continuous everything means using the same practices in development as in operations and sustainment. DevOps practices are founded on automation for continuous integration, delivery and deployment, and operations and sustainment. The approach to DevOps in this document is to build systems to be secure and verifiable from the very beginning. Risk management, quality assurance (QA), and testing are the practices that make accelerated velocity and continuous delivery possible.

The continuous delivery, testing, and QA practices are shifted left (earlier in the workflow) to be planned and executed at the same time as design and development. For example, in DevOps, automated tests are built along with the product from inception. Most efforts begin with effective reviews of requirements, test strategies, and coding standards. Common methods include test-driven development (TDD), automatic code scanning (on build), automated regression testing, and functional and non-functional (e.g., performance) testing. Continuous QA and testing are essential in any DevOps-centric effort. Reducing rework and waste contributes to the achievement of improved velocities, a hallmark of well-implemented DevOps.

Similarly, information security cannot be tacked on to the end of a development effort. The DevOps view of security is sometimes referred to as DevSecOps, but in reality, there is no DevOps without a continuous focus on security. This includes building systems to be secure from the very beginning of the systems and applications life cycle and continuing throughout their life cycle, including code that is deemed ready to be safely deprecated.

Left-shift is particularly valuable for improving the reliability of methods for production software release and deployment. DevOps transforms a common practice in which developers fully control sandbox, development, and initial test environments, with more rigorous release procedures applied to production. This approach can cause delays and integration and testing issues if toolsets, training, support, and procedures for deployment to production are different than those used in the lower environments. In DevOps, “left-shifting” of deployment procedures means lower-risk deployment using the same methods for all environments in the continuous delivery pipeline.

To accomplish continuous delivery more securely and reliably, many firms have turned away from traditional monolithic, sequential, and mostly manual development and operational approaches to one that integrates an ever-growing number of market-proven external solution components (i.e., frameworks, libraries, application programming interfaces [APIs], and software as service solutions) with a more manageable set of targeted custom solution components. There has also been a significant shift to cloud-based hosting that can easily and efficiently scale up and down as needed to satisfy the dynamic load demands of users. To enable this, DevOps requires the use of tailored processes and specialized pipeline tools able to leverage automation wherever practicable, across the entire system’s life cycle.

5.2.4 Systems thinking

Systems thinking counters a myopic approach of utilizing specialists—such as networking professionals, database administrators, and systems administrators—who rarely communicate with either the development or operations teams and lack understanding of the system as a whole. In DevOps, taking a comprehensive view encourages technology professionals to fully understand the system from end to end. Systems thinking can enable resolution of complex and emergent problems that are not easily traceable to a single

flaw. Systems thinking should apply to a consistent architecture for the enterprise tools used for DevOps as well as to the system under development.

DevOps systems and their interfacing systems, constituent systems, and enabling systems, including human participants, form complex ecosystems. Systems thinking in DevOps involves integrating the widely differing perspectives and notions of value and priorities across the stakeholders. Consistent systems thinking among DevOps stakeholders can be challenging when there are differing or even incompatible philosophies, policies, procedures, activities, tasks, and tooling.

Efforts to bolt on security, reliability, and privacy solutions to existing operational systems seldom work. Establishing procedures and approaches for these areas should be addressed as early as possible in the systems life cycle to benefit from the active involvement of all stakeholders. To achieve this requires the creation of shared vocabularies, models, and processes so that the impact and consequences of changes can be quickly and effectively assessed, evaluated, and used to drive decisions and implement treatments. Communication and collaboration across stakeholders are critical components of an overarching, ecosystem-wide risk management process.

Systems thinking is especially valuable when deployment, operations, and post-deployment support activities are transitioned from one organization to another (e.g., internationally deployed systems and deep space missions).

5.3 DevOps and organizational culture

5.3.1 Leadership

While DevOps involves cooperation at all levels of an organization, it is most successful when leadership is viewed internally and externally as fully supporting the letter and the spirit of the DevOps policies, principles, and practices. When those in leadership roles exert their authority and influence, the entire DevOps team is better able to interact with stakeholders; uphold the power of organizational procedures; sustain a sophisticated, complex, and effective pipeline; and engage capable human resources.

Addressing DevOps vulnerabilities requires commitment to investments to establish, sustain, and improve capabilities so that the ecosystem remains aligned with evolving policies, processes, mechanisms, practices, and tools. Without clear and lasting support from and continued active engagement of leadership, the benefits of investments in procedures and tools often fade and disappear. It takes time, the long-term leadership commitment of considerable resources, and a level of dedication to fully deploy and realize the continuous improvement of the DevOps principles, practices, and processes described in this document.

DevOps leadership benefits from the use of transparent, information-driven, risk-based decision-making at all levels. To achieve velocity and other quality goals for DevOps, proper utilization of proven forms of automation depends on the involvement of humans receiving and acting on statistically valid information. These mechanisms and processes are particularly challenging to establish, utilize, sustain, and improve due to the differing baseline cultures of the stakeholders. Proactive and engaged leadership can set the example for stakeholders in recognizing the value and applicability of shared data-driven decisions. When the analysis of information shows statistically significant linkages, use of automation can detect and resolve issues within carefully established bounds. This proactive approach can free leaders and other key stakeholders to focus on innovative and strategic opportunities and risks as well as investments to stay ahead of the competition and, if at all possible, the customers.

Leadership benefits from clear and timely procedures for escalation and dynamic information-driven risk management and issue resolution. Decision-making at all levels empowers the team and the organizational leaders with the capability to handle emerging opportunities and risks inherent in the DevOps ecosystem. This approach helps avoid the overload and burnout of leadership and other personnel by involving,

empowering, and authorizing the right people with the right capabilities at the right time with the right information to do the right thing.

5.3.2 Organizational structure and dynamics

This document is intended for use by organizations of any size, particularly for organizations that have requirements for regulatory and audit compliance, such as segregation of duties.

For some very small entities (e.g., fewer than ten employees), it can be possible to operate without formal organizational structures. The same team "owns the code" from design creation through to daily operations, and a DevOps approach is inherent. As the system grows in complexity, however, new frameworks, libraries, APIs, automated tools, and more sophisticated processes can be utilized to enhance the organization's capabilities. As a result, new stakeholders are added and become key to system success. The small organization becomes part of a much more complex ecosystem, even though the number of developers in the organization has not increased.

The dynamics of an organization impact the approach to a successful DevOps transformation. For example, startups and small (rapidly growing) financial services firms often have very different dynamics compared to large financial services such as international banks and global enterprise organizations. The organizational dynamics of medical and pharmaceutical companies are also quite different than those found in a government contractor. The approach to a successful DevOps transformation should be aligned to the organizational dynamics, internal and external competitive forces, and constraints on the organization.

DevOps is as much a cultural approach as it is a change in specific organizational and technical procedures. To achieve the quality and velocity goals that make DevOps attractive, people with relevant knowledge, skills, and abilities need to have the information, tools, resources, and authority to do what is required. Use of DevOps seeks to eliminate the negative impact of organizational silos while maintaining organizational procedures that are required for compliance. DevOps can be practiced using many organizational structures where there is a commitment to collaboration and continuous improvement, such as continuous workforce training, skill acquisition, capability development, and career progression. The commitment of the organizational leaders is particularly valuable to enable and sustain this cultural approach.

DevOps serves teams that work at a consistent sustainable pace with consideration for work-life balance. This continuous workflow contrasts to the expectation that leading up to a release, developers and operations staff have to work long hours and make extraordinary efforts to achieve difficult—even impossible—goals. While the "work-hard/play-hard" culture still exists, there is a growing recognition that stressed and exhausted employees make mistakes, which the team once again goes through heroic efforts to solve. In contrast, DevOps works with many smaller, simpler, and fully automated deployments which can be executed in the normal workday with no perceptible impact to the end user.

5.3.3 Effective communication and collaboration

While internal and external forces can have a tremendous impact on any organization, the way in which its members interact is even more consequential. DevOps values effective and collaborative communication and consequently thrives in organizations where transparency and collaboration are expected and enjoyed. Transparency prioritizes ease of visibility, availability, reachability, and accessibility of information and actions between entities, people, or systems. Transparency enables communications, accountability, and data-driven decision-making.

DevOps is often characterized as a practice to help improve communication and collaboration between stakeholders who have different goals and objectives and may even belong to different internal and external organizations. Developers and operations teams should collaborate, especially when a separate operations team is required to maintain control of the production systems.

When dealing with new tools supporting "continuous everything," unforeseen issues can arise. DevOps values communication and feedback loops, allowing the organization to experiment with varying approaches. A feedback loop is a virtuous cycle where action produces information which is used in future action to improve results. Team members regularly communicate what is going well and what needs to be improved. DevOps helps implement systems and software life cycle processes with regular checkpoints and rapid course correction when warranted.

5.3.4 Learning organizations and knowledge management

DevOps organizations should embrace iterative or continuous learning, sharing, persisting, and practicing the application of knowledge and best practices. Organizational structures and systems which help to retain, refresh, and leverage technical knowledge and lessons learned are fundamental, as is a culture which enables and encourages its members to share their knowledge and experience. Reflection on the value of lessons learned and their reuse are particularly important when a system or project transitions to other contexts or is handed off to other teams. Knowledge-sharing organizations are most often high-achieving organizations which foster high-performance cross-functional teams. Training, practice, and knowledge sharing for productive and consistent use of enterprise processes toolsets are often key factors in successful enterprise DevOps adoption.

5.3.5 Adaptation, resilience, and organizational change

Central to realizing the benefits of DevOps is successful adaptation of its principles and techniques to the needs of the organization. Adaption is particularly important given the dynamic and competitive nature of DevOps, especially critical in times of societal and domain stresses. The adaptation of DevOps processes to perform at velocity with periods of acceleration is enhanced with experienced-based continuous improvement, a growing reliance on automation, and enlightened risk management. Effective adaptation of DevOps continuously invests in transforming work into automated services that improve velocity, support sporadic acceleration, and enhance other quality attributes, thereby allowing leadership and stakeholders to be more strategic.

While success in a highly competitive domain depends on acceleration and continuously achieving a high level of performance, implementing DevOps involves actually leveraging the available resources with their varying experience levels and skill sets. Exerting extreme efforts whenever mission success, lives, and crucial infrastructures are at risk is not a sustainable tempo in the long term. Ensuring long-term success depends on effectively establishing, utilizing, and improving human resources to gather statistically-meaningful information, analyze the information, and make decisions within mutually-agreed risk tolerances.

Changing an organization to apply DevOps begins with an assessment of current practices to help identify risks and opportunities, and set the direction and priorities for improving practices. Assessment of the existing communications patterns and software development, delivery, operations, and sustainment practices across the ecosystem helps the team to compare as-is and to-be practices and to identify and address gaps. Identifying organizational risks, problems, and issues affecting product development, delivery, operations, and sustainment can act as a catalyst to successful DevOps adoption.

Bringing about organizational change can be a challenging endeavor, especially when external stakeholders need to change their culture or practices. Resilient organizations aim to enable rapid innovation while also minimizing issues due to service interruption, due to the pragmatic cost of quality. Many change agents focus on implementing best practices within their own organizational unit. While it is common to start small—perhaps within one organizational unit sponsoring the DevOps transformation and its ecosystem, it is also essential to consider how these practices may be instantiated consistently throughout the organization. The broader goal is to promote collaboration for systems and processes which enable DevOps across the entire organization.

Understanding the distinct perspectives of development, operations, and the many other key stakeholders involved in the systems and applications life cycle effort allows DevOps to take hold. Giving equal consideration to other stakeholders, including information security, quality assurance, testing, users, suppliers, and partners can represent a considerable change in organizational dynamics and culture. The involvement of these stakeholders is needed to implement left-shift changes in organizational practice.

5.4 DevOps and life cycle processes

DevOps is sometimes viewed as being mostly focused on systems and applications deployment, and thus applicable mainly at the end of the life cycle. In practice, DevOps is a full life cycle endeavor which gives equal consideration to each stage. DevOps is a set of principles and practices which enable better communication and collaboration between relevant stakeholders for the purpose of specifying, developing, continuously improving, and operating software and systems products and services. It is not just a matter of technical practices affecting other life cycle processes.

Teams using DevOps typically start a systems or applications effort by creating a continuous delivery pipeline (set of tools and procedures) that takes the code from the source code management system and automates the complete application build, package, deployment (including transitions to other environments), operations, and sustainment workflow. Contributors often start with a simple program, write the pipeline, and then iteratively (and rapidly) develop their code. In development, multiple teams often integrate code continuously, automatically deliver the code to a test automation framework, and on to subsequent workflow participants. Transitions to other contexts in the workflow can require special enhancements to the workflow (e.g., simulation-based training or shadow operations) so that quality and velocity are not compromised during and after any transition.

DevOps is suitable for most life cycle process models, and particularly appropriate when teams adopt agile methodologies. DevOps can be just as valuable in an iterative waterfall approach.

Clause 6 of this document details the outcomes and tasks involved in applying DevOps to each of the life cycle processes identified in ISO/IEC/IEEE 12207:2017 [B14] and in ISO/IEC/IEEE 15288:2015 [B16]. The order in which the processes, activities, and tasks are presented in this document does not imply the order in which the processes are to be implemented or performed. The processes can be performed iteratively and concurrently with other processes. Within legal and regulatory constraints (e.g., segregation of duties), the same person or group can perform multiple processes, and a process can be performed by people in different organizations.

6. Relation of software life cycle processes to DevOps

6.1 Agreement processes

This subclause specifies the requirements related to DevOps for agreements with organizational entities external and internal to the organization.

The Agreement processes consist of the following:

a) Acquisition process—used by organizations for acquiring products or services;

b) Supply process—used by organizations for supplying products or services.

These processes define the activities necessary to establish an agreement between two organizations. If the Acquisition process is invoked, it provides the means for conducting business with a supplier. This may include products that are supplied for use as an operational software system, services in support of operational activities, software elements of a system, or elements of a software system being provided by a supplier. If the Supply process is invoked, it provides the means for an agreement in which the result is a product or service that is provided to the acquirer. (ISO/IEC/IEEE 12207:2017, 6.1.)

NOTE—Throughout Clause 6, boxed text is used to identify direct quotations regarding process groups and process purposes from ISO/IEC/IEEE 12207:2017 [B14].

The Agreement process shall establish a strategy for conducting the acquisition. The Agreement process for secure and reliable DevOps includes a number of outputs beginning with the General Work Statement, which is agreed by the acquirer and the supplier. The agreement includes the more detailed specification referred to as the service level agreement (SLA). The SLA formalizes the agreement between the supplier and the acquirer for the delivery of services within defined criteria and for a specified cost and time-frame.

The Agreement shall include requirements so that the service level agreement and operations level agreement (OLA) work in unison. Organizations can act simultaneously or successively as both acquirers and suppliers of services. The operating level agreement (OLA) shall be put in place to establish the linkages and responsibilities between IT service provider providers whether they are internal or external. The OLA should include software and hardware licensing and maintenance agreement information with any underlying client and vendor contracts. The OLA shall include data to establish, verify, and maintain the agreement for the integrity of the services. The OLA shall establish mechanisms and criteria for the evaluation of the quality, performance, and security of the service. Where the supplier and acquirer are within the same organization, they may still use the Agreement process for consistency throughout all stages of the life cycle.

6.1.1 Acquisition process

6.1.1.1 Purpose of the process

The purpose of the Acquisition process is to obtain a product or service in accordance with the acquirer's requirements. (ISO/IEC/IEEE 12207:2017, 6.1.1.1)

The agreement shall specify supplier responsibility for defining functional and non-functional requirements that have an impact on quality. At the early stages of negotiation, the requirements may be high-level. Agreement wording that reinforces the importance of deriving, reviewing, and testing functional and non-functional requirements is necessary for successful DevOps implementation when the product is delivered. Non-functional requirements such as capacity, security, performance, and maintainability address concerns that become apparent at later stages of implementation when the product is required to fit into the target test and production environments for final shakedown, acceptance, and long-term use by internal or external customers.

Example: The following specifies supplier responsibilities for security: "The supplier shall define capabilities to build in privacy and security by design, meaning any data collected must be securely stored, processed, and use the minimum data necessary to deliver the services and service-related data. It also means that controls to secure the product or service shall be explicitly stated. The Supply process shall define the procedures for gathering and analysis of contractual documents related to information security of the product or service, which can include the service level agreement and the operations level agreement.

The process shall include a description of the most current assurance of the reliability of the service provider's information protection program and the certainty of the security controls supplied by the service provider."

DevOps goals frequently require a higher level of process maturity or capability, as evidenced by automated procedures to meet organizational and stakeholder goals and objectives effectively. Vendors from which acquisitions are being integrated may need to demonstrate their dependability to support DevOps-based processes.

6.1.1.2 Outcomes

As a result of the successful implementation of the Acquisition process, achievement of the following outcomes shall be demonstrable:

a) The risk associated with the proposed supplier is understood.

b) Service level agreements (SLAs), service level objectives (SLOs), and service level measures and indicators are agreed, consistent with DevOps pipelines (capacity).

c) Agreements include completion criteria ("what does 'done' look like") at the complete system level, for each level of work, and for each deliverable.

6.1.1.3 Activities and tasks

The organization shall implement the following activity and tasks in accordance with applicable organization policies and procedures for DevOps with respect to the Acquisition process:

a) Evaluate the capability or maturity of the supplier.
   1) Determine the appropriate models for evaluation.
   2) Review evidence of appropriate process capabilities, such as:
      i) Vendor policies
      ii) Results of penetration testing
      iii) Frequency of applying security patches
      iv) Existing processes and practices
      v) Verification and Validation activities
      vi) User acceptance test plans
      vii) Proof that the software is working as required (functional configuration audit [FCA])
      viii) Test summary reports as evidence of testing completed

b) Align the commonly required service level agreement (SLA) of the acquirers and suppliers, avoiding jeopardizing the dependent providers' and consumers' functional SLA.

6.1.2 Supply process

6.1.2.1 Purpose of the process

The purpose of the Supply process is to provide an acquirer with a product or service that meets agreed requirements. (ISO/IEC/IEEE 12207:2017, 6.1.2.1)

The Supply process shall define the scope of responsibility which the service provider can accept.

The Supply process shall include a detailed description of processes and procedures used to help the product or service meet its intended use and be fit for use. This description should include supplier services, such as vendor-specific tools, and dependencies management (framework/library/open source or vendor-specific tools) for both build and runtime dependencies. Delivered components shall be specified with their version, vendor origin, and licensing. Use of cryptographic tools shall be identified.

NOTE—ISO/IEC 19770 (all parts) [B6] defines a management system for IT asset identification.

The supplier shall define capabilities for the secure storage, process, and delivery of services and service-related data. The Supply process shall define the procedures for gathering and analysis of contractual documents related to information security of the service, which can include the service level agreement and the operations level agreement. The process shall include a description of the most current assurance of the reliability of the service provider's protection of information and the certainty of the security controls of the service provider.

6.1.2.2 Outcomes

As a result of the successful implementation of the Supply process, achievement of either of the following outcomes shall be demonstrable:

a) The supplier has the capabilities to provide specified products and service.

NOTE 1—Supplier demonstrations can include descriptions of the suppliers' existing policies and processes, including secure development, the existing security of the product, environment management, and related protocols.

NOTE 2—Records of assessments can be provided to support the description of the product, processes, and environment.

NOTE 3—ISO 27034-1 [B10] and IEC 62443-4-1 [B2] provide additional details on application security.

b) Product acceptance criteria are defined.

6.1.2.3 Activities and tasks

The organization shall implement the following activities and tasks in accordance with applicable organization policies and procedures for DevOps with respect to the Supply process:

a) Provide evidence of capabilities in QA reporting, testing, and governance as related to the agreed work.

b) Provide evidence of capabilities in the following Security areas:

1) Roles and responsibilities to cover security activities
2) Security requirements specification
3) Secure by design
4) Secure implementation
5) Security guidelines
6) Security update management
7) Penetration testing
8) Management of security-related issues in threat modeling, secure coding, and penetration testing

9) Security review

10) Functional test of fulfilling security requirements

c) Provide evidence of capabilities in the following Verification areas:

1) Test strategies

2) Automated code review

3) Static analysis upon check-in

6.2 Organizational Project-Enabling processes

The Organizational Project-Enabling processes help ensure the organization's capability to acquire and supply products or services through the initiation, support and control of projects. These processes provide resources and infrastructure necessary to support projects and help ensure the satisfaction of organizational objectives and established agreements. They are not intended to be a comprehensive set of business processes that enable strategic management of the organization's business. (ISO/IEC/IEEE 12207:2017)

The Organizational Project-Enabling processes shall align with DevOps principles and practices such as including relevant stakeholders. The Organizational Project-Enabling processes shall be integrated into the DevOps pipeline and measurements related to organizational objectives should be gathered and monitored across the life cycle.

6.2.1 Life Cycle Model Management process

6.2.1.1 Purpose of the process

The purpose of the Life Cycle Model Management process is to define, maintain, and assure availability of policies, life cycle processes, life cycle models, and procedures for use by the organization with respect to the scope of this document.

This process provides life cycle policies, processes, models, and procedures that are consistent with the organization's objectives, that are defined, adapted, improved and maintained to support individual project needs within the context of the organization, and that are capable of being applied using effective, proven methods and tools. (ISO/IEC/IEEE 12207:2017, 6.2.1.1)

DevOps demands higher levels of integration, collaboration, automation, and established feedback systems for continuous improvement into the life cycle process model. Higher-level process capabilities specifically bring security into prominence in the activities and tasks.

6.2.1.2 Outcomes

As a result of successful implementation of the Life Cycle Management process, achievement of the following outcomes shall be demonstrable:

a) Specific policies and procedures for managing the life cycle are agreed and implemented, consistent with organizational objectives.

b) Stakeholders are identified and communication is facilitated throughout all life cycle stages.

c) Security is integrated throughout all stages in the life cycle.

d) Life cycle processes and process models and DevOps policies and procedures are implemented and integrated throughout life cycle stages.

NOTE—To the extent policies and procedures of the life cycle model and management processes are increasingly automated and auto-refined, they also enable increased speed of change in relation to external factors, such as business need.

6.2.1.3 Activities and tasks

The organization shall implement the following activities and tasks in accordance with applicable organization policies and procedures for DevOps with respect to the Life Cycle Model Management process:

a) Establish Life Cycle Model Management in alignment with the following:
   1) Principles and practices, such as including relevant stakeholders
   2) Continuous data monitoring
   3) Project needs and goals
   4) Life cycle model for a given project
   5) Business activities
   6) Organizational policies and procedures for governance, risk, and compliance

b) Align Life Cycle Model Management with the following security considerations:
   1) Accountability throughout the DevOps processes
   2) Continuous tracking and evaluation of security needs
   3) Continuous monitoring and measurement of security performance
   4) Continuous improvement of security performance
   5) Security communication throughout the life cycle

6.2.2 Infrastructure Management process

6.2.2.1 Purpose of the process

The purpose of the Infrastructure Management process is to provide the enabling infrastructure and services to projects to support organization and project objectives throughout the life cycle.

This process defines, provides and maintains the facilities, tools, and communications and information technology assets needed for the organization's business with respect to the scope of this document. (ISO/IEC/IEEE 12207:2017, 6.2.2.1)

In DevOps, the complexity of the systems, the diversity of stakeholders, and the need for velocity make it a non-trivial endeavor to provide an appropriate infrastructure (ecosystem). DevOps goals frequently involve a higher level of process capabilities to meet organizational and stakeholder goals and objectives effectively, and increased infrastructure needs to support dependable, automated DevOps-based processes. The DevOps use of pipelines with automation requires a high degree of coordination, collaboration, and issue resolution across the ecosystem. DevOps principles and practices should apply a set of data-driven and risk-based policies and procedures for leadership, management, operation, and continuous improvement.

Enabling velocity requires a continuous focus on data-driven efforts to identify and effectively remove waste from the process and prevent its recurrence. Design, implementation, management, and improvement of the pipeline systems and processes involves effective risk management. A risk-based ecosystem concept of operations should explicitly address how communication, collaboration, oversight, and improvement will be implemented using various DevOps pipeline roles, responsibilities, tools, and processes.

Integrating the ecosystem is typically an iterative process with design changes and accommodations to differing risk appetites and tolerances of stakeholders. Assessing, analyzing, designing, implementing, refining, and improving the ecosystem continues throughout its life cycle. Automated DevOps approaches often can take advantage of virtual or cloud infrastructure, allowing additional environments to be provisioned on demand for short-term testing or continued use. Use of software for rapidly provisioning infrastructure is referred to as *infrastructure as code (IaC)*.

6.2.2.2 Outcomes

As a result of the successful implementation of the Infrastructure Management process, achievement of the following outcomes shall be demonstrable:

a) Network infrastructure to support DevOps is provided.

b) Continuous communication mechanisms to support DevOps are implemented throughout the life cycle.

c) Security services are implemented throughout the life cycle.

NOTE—Relevant security standards include ISO/IEC 27001 [B8], ISO/IEC 27033-1 [B9], and ISO/IEC 27034-1 [B10].

d) Procedures for environment replication, e.g., for the evaluation of integrated systems or maintenance work, are established early in the systems life cycle.

e) Infrastructure as code (IaC) is supported.

6.2.2.3 Activities and tasks

The organization shall implement the following activities in accordance with applicable organization policies and procedures for DevOps with respect to the Infrastructure Management process:

a) Plan for environment, infrastructure, and resource needs (e.g., for budgeting, scheduling, change management, or risk management).

b) Specify non-functional requirements for characteristics including reliability, security, dependability, or scalability, according to the context of the actual infrastructure services.

c) Provide for automation and configuration management (CM) tools within the information technology portfolio of the organization.

d) Select planning and management tools with low overhead and high visibility.

e) Implement software development tools, including version control tools and document automation scripts dependencies.

f) Support the selected development approach with the infrastructure.

g) Implement an infrastructure that supports security activities, a test environment, and other verification and validation activities.

h) Enable infrastructure as code (IaC) with support of security and compliance analyses.

i) Maintain security and compliance analysis of IaC.

j) Support infrastructure provisioning through automated mechanisms with machine-readable definition files.

6.2.3 Portfolio Management process

6.2.3.1 Purpose of the process

The purpose of the Portfolio Management process is to initiate and sustain necessary, sufficient and suitable projects in order to meet the strategic objectives of the organization.

This process commits the investment of adequate organization funding and resources, and sanctions the authorities needed to establish selected projects. It performs continued assessment of projects to confirm they justify, or can be redirected to justify, continued investment. (ISO/IEC/IEEE 12207:2017, 6.2.3.1)

In DevOps, a systems perspective shall be taken for Portfolio Management. The organization's IT Portfolio is composed of systems for which projects are established to develop, enhance, maintain, and eventually retire those systems in order to meet the IT needs of the organization as well as the needs of the end user. The information to inform portfolio decisions should be collected in a frequent, automated, and collaborative manner.

To achieve the Portfolio Management outcomes, DevOps processes should operate at a higher level of process capability to meet organizational and stakeholder goals and objectives effectively.

6.2.3.2 Outcomes

As a result of the successful implementation of the Portfolio Management process, achievement of the following outcomes shall be demonstrable:

a) A current functional view of the products, systems, and services offered within the portfolio is maintained.

b) The disposition of systems (install, modernize, remove, upgrade, maintain) is rationalized against performance, capacity, security, compliance, and risk concerns.

6.2.3.3 Activities and tasks

The organization shall implement the following activities in accordance with applicable organization policies and procedures for DevOps with respect to the Portfolio Management process:

a) Create organizational policies regarding resiliency, continuity of operations, security, compliance, and risk.

b) Implement projects to sustain business risk tolerance for resiliency, security, and regulatory compliance.

c) Collaborate regularly on the disposition of critical systems.

d) Commit and prioritize the resources needed to develop the features and tools for building and sustaining the continuous processes, such as continuous integration/continuous delivery (CI/CD).

e) Perform portfolio level reporting in a simplified, non-technical manner that clearly explains the current state of security, compliance, and risk across critical business functions.

6.2.4 Human Resource Management process

6.2.4.1 Purpose of the process

The purpose of the Human Resource Management process is to provide the organization with necessary human resources and to maintain their competencies, consistent with business needs.

This process provides a supply of skilled and experienced personnel qualified to perform life cycle processes to achieve organization, project, and stakeholder objectives. (ISO/IEC/IEEE 12207:2017, 6.2.4.1)

Strategic planning for Human Resources is evidence-based using historical demands and industrial trends. In DevOps, teams employ continuous improvement and left-shifting, continuous, cross-functional communication and collaboration, and knowledge-sharing among stakeholders to reduce the negative impact of organizational silos. This implies that personnel have a cultural flexibility and capacity to quickly evaluate, understand, and successfully adopt new technologies, methods, tools, and behaviors while intentionally removing the obsolete ones. Team support during early use of recently learned skills can improve job performance. DevOps requires tool-usage skills that are closely tied to the pipeline.

Human Resource Management supports onboarding, practice, coaching, and mentoring in DevOps procedures. Those responsible for the creation of work products should be properly trained and competent. From the perspective of the individual, learning about the organization's approach to DevOps occurs during onboarding and continued micro-learning includes the product functionality, domain engineering processes, and integration processes. The organization shall implement learning activities to support organizational specific activities and tasks, e.g., commercial-off-the-shelf (COTS), open source software (OSS), third-party, and custom development and integration.

The following DevOps skills are part of competency in this domain. The expected level of competence depends on the project roles and individual responsibilities.

a) Soft skills:
    1) Team-oriented working skills
    2) Written and verbal communication skills
    3) Reasoning and problem-solving skills
    4) Ability to work effectively and manage time without supervision
    5) Ability to respond to customer requirements
    6) Ability to multi-task, prioritize, and take action

b) Technical skills:
    1) Understanding of security issues affecting computer systems, networks, databases, and applications
    2) Deploying and handling environments in cloud platforms
    3) Utilizing CM tools
    4) Administering operating systems
    5) Setting up continuous integration and continuous delivery
    6) Collecting and analyzing logs
    7) Security and performance monitoring and tuning
    8) Building and managing web frameworks

9) Releasing deployment packages
10) Maintaining repositories
11) Automating systems and tasks
12) Performing risk, performance, and security assessments
13) Reviewing and developing performance and capacity plans (operational capacity based on load requirements)

Stakeholder and team dynamics can involve higher-level capabilities and roles to achieve the benefits derived from DevOps, such as efficient automation. DevOps goals frequently require a higher level of process maturity to meet organizational and stakeholder goals and objectives effectively, and human resources provides the higher-level skills to support DevOps-based processes.

6.2.4.2 Outcomes

As a result of the successful implementation of the Human Resource Management process, achievement of the following outcomes shall be demonstrable:

a) Competencies, including evidence of flexibility required by the various roles and responsibilities for projects, programs, or systems are identified, documented, and used in the human resource selection process.
b) Policies on diversity (e.g., neurodiversity, skills, gender, background) leverage a variety of perspectives and skills to drive innovation.
c) Human resource requirements and authorization to hire or transfer are provided early enough for onboarding, practice, socialization, and if required, certifications, licenses, and clearances.
d) Competencies of personnel are collaboratively developed, monitored, managed, and maintained or enhanced, using evidence-driven methods, in alignment with the organization's strategic and operational plans so that pipelines can be effectively managed.
e) Personnel without the requisite proven competencies are provided management support, supervision, technical support, and assistance during an effective learning experience.
f) Conflicts in multi-project resource demands are resolved.
g) A positive work environment and culture is sustained that proactively and continuously promotes productivity, productive relationships, and continuity for individual resources and project teams.

6.2.4.3 Activities and tasks

The organization shall implement the following activities and tasks in accordance with applicable organization policies and procedures for DevOps with respect to the Human Resource Management process:

a) **Identify needed competencies.** This activity consists of the following tasks:
   1) Based on historical data, plans, and trends, identify and document the competencies (e.g., knowledge, skills, abilities, and team behaviors) needed for current and expected projects, programs, and systems, to facilitate attracting, hiring, transferring, and onboarding staff.

      NOTE—Development needs align with the organization's strategic and operational plans.

   2) Identify, record, and actively manage the key competencies of personnel using evidence-based, data-driven methods.

b) **Develop proven competencies.** This activity consists of the following tasks:

1) Establish and maintain strategic and operational plans as needed to develop and sustain a pool of proven competent personnel.

NOTE—This plan includes types and levels of training (including knowledge and skills development, practice, and needed certifications), categories of personnel, schedules, and personnel requirements. Training includes onboarding and upskilling.

2) Obtain, or develop and maintain, needed competency development and sustainment resources with special attention given to education or mentoring resources and effective role models.

NOTE—These resources can include, for example, knowledge, skill, ability, and team behavior development materials, courses, practices, and assessment tools (self-assessment tools identical to those used by management) that are developed by the organization or external parties who have tailored those materials, tools, and processes for the required organizational and operational contexts, as well as tailored training courses that are available from external suppliers and tailored computer-based instruction.

3) Provide timely planned competency development and demonstration activities adapted to the contexts of the work to be performed.

4) Enable micro-learning in support of project knowledge acquisition.

5) Maintain records of the effectiveness of development activities as the basis for data-driven continuous improvement.

c) **Acquire and provide proven competencies.** This activity consists of the following tasks:

NOTE—Proven competencies include the recruitment, retention, and development of personnel with experience levels and competencies necessary to properly staff projects, programs, and systems. Appropriately staff assessment and reviews and practice supervision, coaching, and mentoring. It implies that personnel have the proficiencies, motivations, and abilities to work in specified cross-functional team environments, as well as flexibility for retraining, reassignment, or reallocation due to changes in context.

1) Plan for the continuity of productivity and resources across the organization.

2) Engage in planning to resolve deficits in personnel and obtain adequate and timely personnel, resources, management, and leadership to implement the plan.

NOTE—Resource planning can include resources across key stakeholders in the ecosystem for the product.

3) Maintain and manage the pool of proven-competent, diverse personnel necessary to staff ongoing and planned projects, programs, and systems.

4) Using effective, evidence-based, data-driven risk management practices, make task assignments based on documented staff requirements.

5) Motivate personnel through proven effective career development opportunities and reward mechanisms.

6) Control multi-project management interfaces to resolve personnel conflicts.

NOTE—Conflicts of capacity can occur in organizational infrastructure and supporting services and personnel resources among ongoing projects, programs, and systems, or from people being over-committed.

7) Support the process of continuous validation of each individual's current DevOps skills and capabilities.

d) **Enable the organization to proactively create, sustain, and adapt a positive work environment and culture and build a blameless culture and risk-taking mindset.**

6.2.5 Quality Management process

6.2.5.1 Purpose of the process

The purpose of the Quality Management process is to assure that products, services and implementations of the quality management process meet organizational and project quality objectives and achieve customer satisfaction. (ISO/IEC/IEEE 12207:2017, 6.2.5.1)

A DevOps mindset changes the approach to the Quality Management (QM) process by addressing the following concerns:

a) Achieving the customer's objectives, and in turn, the business or mission goals and values
b) Accelerating delivery through continuous QM
c) Collecting and sharing evidence to make informed decisions and achieve customer satisfaction

Some requirements handled by QM derive from compliance to laws and regulations. QM applies a systematic approach to managing the processes that meet customer and regulatory requirements.

Quality means meeting customer requirements. Failure of a system to meet both requirements and expectations results in failure to achieve business goals. DevOps goals frequently require a higher level of process maturity to meet organizational and stakeholder goals and objectives effectively. Therefore, quality is not the responsibility of one person or entity alone, but rather is an outcome of coordinated, systematic, and planned activities at a high level of process capability. Distributed and highly complex systems are a result of many contributors. Accountability is distributed to team members working in conjunction with stakeholders or their representatives throughout the system life cycle.

A lack of visibility into product and service areas often results in a lack of quality, where defects and vulnerabilities can be introduced, such as with handoffs and changes. In DevOps, silos of information should be avoided because they impede the decision process. Instead, information should be made available to those who are or may be affected by the decision, including stakeholders. Therefore, quality-related planning, decisions, execution, and improvement activities are aligned with the organizational quality and risk policies and made transparent to stakeholders.

NOTE—Removing the adverse impacts of information silos does not contradict legal or regulatory requirements for compartmentalized information, nor does it necessarily involve changes in organizational structure or IT controls required for compliance.

Optimal operation requires continuous process monitoring and impact measurement; thus, the QM process should be embedded into the delivery process. This requires effective communication and collaboration with the team so that feedback loops are fit to various release cadences, technologies, and appetites for early feedback on software quality. In DevOps, automated quality gates serve as feedback loops that support informed decisions and alignment between activities and organizational risk policies. Feedback on software quality should go to the contributor that introduced a change. This feedback cycle can be automatic when integration and testing is continuously measured and monitored.

Thinking backward from operational goals, various technologies and processes contribute to the final timeliness and quality of systems delivered to production. Therefore, the QM process establishes traceability, measurement, monitoring, and improvement practices which automate timely delivery of feedback on quality-related requirements to contributors.

As DevOps seeks to reduce the time to delivery between development and operation of systems, getting the right information to the right stakeholders at the right time becomes increasingly critical. Speed is achieved by learning what feedback is meaningful and reducing that which is not. The QM process identifies

meaningful feedback, which in turn improves decision processes and replaces wasteful and toilsome rework with automated, auditable, and versioned procedures.

In a DevOps mindset, hope should not be a strategy; the alternative is to be informed about system quality and operational health with evidence, typically with key performance indicators that are available to stakeholders in real time. QM implements responses to feedback and improves procedures from the following areas:

- Production systems monitoring and product telemetry
- Actionable verification and validation results before and after release to production
- End-user and resource usage
- Impact analysis results
- Inter- and intra-related dependencies required for system operation
- Configuration changes unrelated to approved deployment tasks
- Security and resilience performance capability

6.2.5.2 Outcomes

As a result of the successful implementation of the Quality Management process, achievement of the following outcomes shall be demonstrable:

a) The quality of service consistently meets SLOs across portfolios and pipelines.

b) Deliverables meet stakeholder requirements prior to production and use.

c) Contributors follow standard processes which meet organizational quality and risk policies.

d) Ongoing monitoring confirms that the QA process is functioning effectively.

6.2.5.3 Activities and tasks

The organization shall implement the following activities and tasks in accordance with applicable organization policies and procedures for DevOps with respect to the Quality Management process:

a) **Apply quality standards.** This activity consists of the following tasks:

   1) Identify or develop quality standards and policies that describe:

      i) Required levels of quality and consistency of service across portfolios and pipelines

      ii) Automated software delivery processes that address separation of concerns

   2) Establish the standard procedures that contributors should follow aligned with organizational quality and risk policies, requirements traceability, and continuous improvement.

   3) Determine that quality measures are in accordance with established standards.

   4) Define key indicators and other measures required by stakeholders to make early development and real time operational decisions.

   5) Work with internal and external stakeholders to harmonize definitions of SLA, SLO (goals), and service level indicator (SLI) (measures).

6) Define functional and non-functional quality criteria that can be automatically evaluated prior to operational usage of deliverables or infrastructure systems.

NOTE 1—Product quality attributes can be identified from the ISO/IEC 25010 [B7] product quality model.

NOTE 2—Key performance indicator (KPI) data on other aspects—such as accessibility, usability, interoperability, reliability, and scalability—should also be considered as information critical to the overall QM process supporting business decisions over delivery and operational readiness.

b) **Implement QM procedures.** This activity consists of the following tasks:

1) Establish QA activities across portfolios and development teams, applying cross-project systems thinking.

2) Develop quality-evidential practices (i.e., testing patterns) in accordance with quality and risk policies.

3) Apply automated monitoring of code to be merged (contributions) to identify those that are out of compliance.

4) Establish patterns of correlation and auditability for changes to deliverables between systems.

5) Establish consistent QM policies and procedures for software development and automated deployment pipelines.

6) Establish methods for requirements traceability and automated verification mechanisms as a map between system design intentions and evidence of operational fulfillment.

Examples: Verification matrices, integrating planning/tasking/execution systems, and auditable contribution logs.

7) Facilitate continuous QM throughout the life cycle stages.

8) Identify and monitor critical non-functional aspects of quality over deliverables and pipelines so that agreed service levels are met: SLA, SLOs (i.e., uptime and availability), and SLI (i.e., latency, throughput, and error rate).

c) **Communicate QM activities.** This activity consists of the following tasks:

1) Pursue improved alignment between design and operational goals by practicing effective communication and collaboration with relevant stakeholders.

2) Integrate continuous integration (CI) and continuous delivery/deployment (CD) reports into software life cycle development platforms (i.e., project dashboard, wiki page).

Examples: Service availability and reliability reports, consumer evaluations and ratings, automated production quality reports, third-party client analyses, internal and external consumer satisfaction reports.

3) Establish methods for collecting and analyzing data on quality and make that data visible to relevant stakeholders, including customers.

4) Collaborate with relevant stakeholders to monitor KPIs and quality metrics.

d) **Manage corrections and lessons learned for quality issues.** This activity consists of the following tasks:

1) Coordinate with operations and development to establish critical non-functional criteria, including acceptance criteria and problem-resolution procedures if quality levels fail to meet their targets.

2) Establish procedures that incorporate continuous production telemetry and user feedback on deliverables and operational systems.

3) Provide longitudinal analysis (a historic view) of quality-related KPIs, covering both automated and manual testing activities, as baseline context for future pre-production defect resolution of post-production issue mitigation.

NOTE—This data can improve both development and operational understanding of system behaviors, context over perceived issues, and evidence of operational health for auditing purposes.

6.2.6 Knowledge Management process

6.2.6.1 Purpose of the process

The purpose of the Knowledge Management process is to create the capability and assets that enable the organization to exploit opportunities to re-apply existing knowledge.

This encompasses knowledge, skills, and knowledge assets, including system elements (ISO/IEC/IEEE 12207:2017, 6.2.6.1).

Knowledge Management creates the pipeline, capability, and assets that enable the organization to implement the core concepts of DevOps. This process encompasses personal knowledge, skills, and knowledge assets, including internally developed, acquired, or contracted frameworks and systems. Knowledge management relies on flexibility to create, persist, share, and apply information and knowledge applicable to the context of DevOps. Knowledge management involves cross-functional communication across teams and stakeholder groups. Knowledge management draws from the lessons learned and conclusions drawn from practice which can be applied to make decisions and take action. Its focus is on the people and their experience and judgement.

Example: Knowledge Management can help staff understand the context of the details to a customer needs analysis report and guide the application of decisions made in response to it.

For DevOps, Knowledge Management focuses on software development, testing, quality, infrastructure, security, and operations, with particular interest in areas such as:

- Establishing the automation of just-in-time processes (development, infrastructure, and operations together)
- Workflow orchestration and scripting
- Developer and infrastructure tool stacks
- Cybersecurity, test, and safety engineering

6.2.6.2 Outcomes

As a result of the successful implementation of the Knowledge Management processes, achievement of the following outcomes shall be demonstrable:

a) A visible, accessible, and shareable knowledge management system is applied to support the successful implementation of DevOps principles and practices.

b) Communication of knowledge occurs across the organization.

c) DevOps teams, stakeholders with domain knowledge, and workforce management collaborate to develop and share knowledge.

d) Teams share consistent and current knowledge of the systems configurations and related dependencies regardless of source (COTS, OSS, third party, custom).

6.2.6.3 Activities and tasks

The organization shall implement the following activities and tasks in accordance with applicable organization policies and procedures for DevOps with respect to the Knowledge Management process:

a) Establish a visible and sharable knowledge management environment. This activity consists of the following tasks:

   1) Assess and manage knowledge assets based on domain, technical skills, and organizational common capabilities.

   2) Throughout the systems and software life cycle management activities, share persistent knowledge of lessons learned and artifacts (i.e., notes, failures, success) in addition to subject matter expert (SME) knowledge among stakeholders.

   3) Continuously improve and update the DevOps policies, processes, methods, tools, and knowledge base that support worker currency.

   4) Secure the learning objects and the knowledge management environment.

   5) Establish periodic review procedures for the knowledge management environment.

b) Enable accessible, continuous learning from knowledge resources. This activity consists of the following tasks:

   1) Facilitate collaboration through communities of interest among stakeholders.

   2) Establish knowledge sharing practices that enable comprehensive left-shift of critical activities.

   3) Provide alternate paths for learning from knowledge base artifacts, including micro-learning, embedded learning, mentoring, shadowing, and technical exchange.

   4) Enable reuse of lessons learned across initiatives.

   5) Share knowledge in such a way that it can be easily consumed and acted upon.

   6) Introduce new team members to the DevOps processes, methods, services, and tools.

   7) Validate lifelong learning and continuous professional development.

   8) Conduct lessons learned and retrospective sessions to refresh the knowledge base and improve systems and software development practices.

6.3 Technical Management processes

The Technical Management processes are used to establish and evolve plans, to execute the plans, to assess actual achievement and progress against the plans, and to control execution through to fulfillment. Individual Technical Management processes may be invoked at any time in the life cycle and at any level in a hierarchy of projects, as required by plans or unforeseen events. The Technical Management processes are applied with a level of rigor and formality that depends on the risk and complexity of the project.

The scope of a technical management process is the technical management of a project or its products, to include the software product or system-of-interest. (ISO/IEC/IEEE 12207:2017)

DevOps brings quantitative and automated visibility to Technical Management in order for the extended team to be aware of how success is defined for the complete system, how each stakeholder participates in that achieving success, and the current progress or issues toward achieving success.

DevOps brings organizational visibility to the Technical Management of system domains that may have traditionally operated as silos.

DevOps brings the capability of checking progress against the plan based on automatically generated artifacts, i.e., capturing data that automatically generates progress reports, rather than having to ask individuals.

DevOps brings on-demand and transparent visibility of system status across life cycles.

DevOps aligns Technical Management processes with Portfolio Management. The information collected can inform decisions in a frequent, accessible, automated, and collaborative manner.

DevOps requires activities and tasks to be broken down and implemented in small enough packages to allow for fast iteration and adjustment for unforeseen events. Plans should be revised as goals or stakeholders' input change to incorporate new ideas, discovered aspects and impacts such as regulatory changes.

6.3.1 Project Planning process

6.3.1.1 Purpose of the process

The purpose of the Project Planning process is to produce and coordinate effective and workable plans.

This process determines the scope of the project management and technical activities, identifies process outputs, tasks and deliverables, establishes schedules for task conduct, including achievement criteria, and required resources to accomplish tasks. This is an ongoing process that continues throughout a project, with regular revisions to plans. (ISO/IEC/IEEE 12207:2017, 6.3.11)

For DevOps the project planning process shall also take into account the outcomes, deliverables, processes, resources, and risks associated with implementing, automating, and supporting DevOps pipelines and continuous feedback mechanisms for a complete system across domain life cycles for all stakeholders. Fast and continuous iteration, planning, and execution necessitate an increased focus on dependencies, blockages, and defects when they arise. Setup, execution, and maintenance work to support QA shall be included in time and resource cost estimates.

Project Planning requires a higher level of process capability to meet organizational and stakeholder goals and objectives effectively, such as demonstrating completeness, transparency, and continuous communication with stakeholders.

6.3.1.2 Outcomes

As a result of the successful implementation of the Project Planning process, achievement of the following outcomes shall be demonstrable:

a) Project planning implements organizational policies (e.g., risk, security, and test management).

b) Project work breakdown facilitates fast iteration and low risk to meet business and technology changes.

c) Project planning is tightly integrated from both a practice and tools perspective, enabling collaborative effort across stakeholder teams as they perform their work.

d) Planning is automated to accommodate fast iterative adjustments, enable plans to evolve, and maintain close alignment across the organization.

e) Project planning enables rapid, frequent, and reliable system delivery.

6.3.1.3 Activities and tasks

The organization shall implement the following activities and tasks in accordance with applicable organization policies and procedures for DevOps with respect to the Project planning process:

a) **Define the project.** This activity consists of the following tasks:

1) Define policies for project planning as agreed with relevant stakeholders.

2) Provide training on the standard planning tools and techniques.

3) Define roles, responsibilities, accountabilities, and authorities for project planning as agreed with stakeholders.

4) Implement visible incentives to promote successful project planning, information flow, and integration across the organization.

b) **Plan the project and its technical management.** This activity consists of the following tasks:

1) Define, integrate, and baseline plans that align with business value.

2) De-emphasize highly structured events and meetings in favor of online shared information and collaboration.

3) Apply the minimum viable product (MVP) approach in planning for deliverables.

4) Enable normalization of frequent change through compact low-dependency scopes, low gates, low overhead, and fast handoffs.

5) Prioritize high return, low risk, and limited duration activities to enable fast iteration.

6) Communicate with stakeholders whether requirements are fixed or emerging.

7) Plan to review risks in relation to finished product due dates and end dates.

8) Incorporate automated feedback loops from stakeholders with enough frequency for agile adjustment of the plan and implementation activities.

9) Estimate and incorporate unplanned work.

10) Provide buffer timeframes to account for the accommodation of unplanned work (incidents, defect cycles) and mechanisms for improvements to be implemented (continuous improvement).

c) **Activate the project.** This activity consists of the following tasks:

1) Communicate a unified message of project, system, and business goals to relevant stakeholders and seek to eliminate technical management information silos among the various stakeholders and teams.

2) Distribute plans continuously to stakeholders in a transparent manner to enable analysis, problem detection, and issue resolution.

NOTE—This task includes collaborating with stakeholders for planning, collecting, communicating, and analyzing information in stakeholder plans to anticipate and avoid blocked work.

3) Commit, fund, and prioritize the technical resources needed to obtain or develop the features, tools, and techniques for building and sustaining project and technology management (e.g., CM, automation, continuous processes (CI, CD), dashboards, critical path planning, kanban).

6.3.2 Project Assessment and Control process

6.3.2.1 Purpose of the process

The purpose of the Project Assessment and Control process is to assess if the plans are aligned and feasible; determine the status of the project, technical and process performance; and direct execution to help ensure that the performance is according to plans and schedules, within projected budgets, to satisfy technical objectives.

This process evaluates, periodically and at major events, the progress and achievements against requirements, plans and overall business objectives. Information is provided for management action when significant variances are detected. This process also includes redirecting the project activities and tasks, as appropriate, to correct identified deviations and variations from other technical management or technical processes. Redirection may include re-planning as appropriate. (ISO/IEC/IEEE 12207:2017, 6.3.2.1)

In DevOps, the Project Assessment and Control processes take into account the outcomes, deliverables, processes, resources, and risks associated with implementing, automating, and supporting DevOps pipelines and continuous feedback mechanisms for a complete system across domain life cycles for stakeholders.

6.3.2.2 Outcomes

As a result of the successful implementation of the Project Assessment and Control processes, achievement of the following outcomes shall be demonstrable:

a) Automation of continuous integration, continuous delivery, continuous testing, and continuous deployment inform the Project Assessment and Control processes to provide insight into the progress and well-being of the project.

b) Project assessment maintains a current and transparent view of the products, systems, and services to the teams and extended stakeholders.

c) Performance measures and assessment results are disseminated continuously to enable analysis, problem detection, traceability, and issue resolution.

d) Corrective action is defined and directed at the problem source when project achievement is not meeting targets.

e) Project tracking, assessment, and control processes are continuously improved as directed by gathered data and project needs.

6.3.2.3 Activities and tasks

The organization shall implement the following activities in accordance with applicable organization policies and procedures for DevOps with respect to the Project Assessment and Control process:

a) Apply automated data collection and decision-making to help identify and resolve dependencies, blocks, and defects promptly, near the point of introduction.

b) Apply consistent project tracking, assessment, and control tools for technical management throughout the organization to support integrated management of cross-project, program, or portfolio dependencies, as well as project tracking and assessment.

c) Present project tracking, assessment, and control information in media such as dashboards so that the information is appropriately technical and easily consumed by stakeholders.

d) Assess the project, focusing on the current state of technical progress, security, compliance, and risk across critical functions.

e) Use feedback from experimentation to improve practices.

f) Analyze and resolve deviations in project performance near the point of introduction, based on alerts at appropriate risk thresholds.

g) Capture records of major dependencies, decisions, risks, and future opportunities, as strategic input for portfolio management. Use resources allocated to an "error budget" to balance technical debt with new work.

h) Adjust allocation of resources from multiple stakeholders to support project plans and objectives.

6.3.3 Decision Management process

6.3.3.1 Purpose of the process

The purpose of the Decision Management process is to provide a structured, analytical framework for objectively identifying, characterizing and evaluating a set of alternatives for a decision at any point in the life cycle and select the most beneficial course of action. (ISO/IEC/IEEE 12207:2017, 6.3.3.1)

For DevOps, a timely and effective decision-making framework involves getting input from those most experienced and informed to identify the alternative(s) and the preferred outcomes for the situation. Decision-making consistent with policies is enabled at the lowest feasible level to resolve issues and respond to requests.

DevOps helps implement effective decision management systems by automating and integrating the collection of product and process information from across the organization and making that information available based on organization or stakeholder information management policy. Automating routine decision-making based on AI and experience can eliminate unnecessary steps, resulting in a lower cost of failure and clear articulation of risk.

6.3.3.2 Outcomes

As a result of the successful implementation of the Decision Management process, achievement of the following outcomes shall be demonstrable:

a) Decision objectives, a preferred course of action, decision rationale, assumptions, and dependencies are identified.

b) Decisions that can be pre-authorized (by relevant stakeholders) and pre-positioned (contingencies) are identified, evaluated, and automated, if possible, or else recorded for future review and re-use.

c) Controls are in place to address regulatory requirements, surface false assumptions, and counter cognitive biases, improving the quality of data for decision-making.

d) Feedback loops and readily available data confirm that the decision-making process is working as anticipated.

6.3.3.3 Activities and tasks

The organization shall implement the following activities and tasks in accordance with applicable organization policies and procedures for DevOps with respect to the Decision Management process:

a) **Prepare for decisions.** This activity consists of the following tasks:

1) Define a decision management strategy.

2) Identify domain models in use and how those models can improve the quality of decisions.

3) Identify the circumstances and need for a decision.

4) Identify relevant stakeholders to be involved in the decision management strategy for each decision.

5) Identify procedures, methodologies, techniques, systems, repositories, and tools associated with the decision-making that make the recorded data available online and in real time.

6) Identify decisions that can be pre-authorized and applied through automated systems.

7) Identify and apply appropriate organizational security and privacy policies to decisions and decision information.

b) **Gather and analyze the decision information.** This activity consists of the following tasks:

1) Identify the data required to enable decision-making.

2) Collect, store, and share data required to enable decision-making through repositories that are online and in real time.

3) Identify the criteria for evaluating alternatives.

4) As feasible, automate data gathering, data analysis, and feedback loops to support decision-making.

c) **Make and manage decisions.** This activity consists of the following tasks:

1) Identify decisions requiring alternative analysis for relevant stakeholders.

2) Select and declare the decision management strategy for each decision.

3) Determine desired outcomes and measurable decision-making criteria.

4) Identify and evaluate alternative courses of action against the criteria, considering contingencies and risks.

5) Determine the preferred alternative for each decision.

6) Record, track, evaluate, and report decisions online and in real time where reasonable and used for continuous improvement.

6.3.4 Risk Management process

6.3.4.1 Purpose of the process

The purpose of the Risk Management process is to identify, analyze, treat and monitor the risks continually.

The Risk Management process is a continual process for systematically addressing risk throughout the life cycle of a system product or service. It can be applied to risks related to the acquisition, development, maintenance or operation of a system. (ISO/IEC/IEEE 12207-2017, 6.3.4.1)

Risk Management shall be planned, implemented iteratively with tightly integrated stakeholder feedback loops, and continuously improved to prevent problems and ensure delivery. Instead of risk management being limited to a static exercise where risks are listed and never revisited, DevOps Risk Management is a dynamic process where the process of risk identification, analysis, and treatment can be automated, with leadership being alerted to risk changes that may affect project success characteristics, such as velocity, variability, volume, value, or variety.

NOTE—The DevOps Risk Management framework is based on ISO/IEC/IEEE 16085 [B19], which is consistent with the general risk management approach of ISO/IEC 31000 [B11]. ISO/IEC TR 33015:2019 [B13] addresses and provides guidance on how to manage the process risks of not achieving process outcomes. IEC 31010 [B1] provides qualitative and quantitative risk assessment techniques and methods to support suitable adaptations in a particular risk context.

In DevOps Risk Management, types of risk interact, such as:

a) Risks from deficiencies in performing the Risk Management process itself, such as not identifying and considering the relationships, interactions, and impacts of risks from organizational, program, project, or system perspectives

b) Risks from one-time or intermittent risk assessment, rather than continual reassessment and mitigation of the risks

DevOps Risk Management is applicable to all organizations and all levels of an organization. The Risk Management process in DevOps can be applied to address the organizational, program, or project consideration of risk. This approach accommodates, for example, organizations pursuing innovation and investment in the pursuit of opportunity and also those where safety and security require a high degree of focus on the negative aspects of risks and their treatments. The organization's risk appetite, or tolerance for risk, affects its balance of risk mitigation and performance activities in the dynamic DevOps environment.

DevOps demands an iterative, continuous, and information-driven approach to risk management, proportional to customer needs and risk tolerances. Risk processes, when properly deployed in DevOps, include the automated identification, analysis, treatment, and monitoring of risk via DevOps tools and pipelines.

DevOps organizations apply risk management to meet stakeholder (customer) expectations for high-quality products, faster turnaround times, and the early introduction of security and audit procedures. For example, enabling parallel implementation and integration processes and reducing the batch size of a delivery can reduce risk. (However, decreasing module size in a software release can generate new architecture-related risks in managing a plethora of smaller modules.) Using techniques for risk treatment, such as regression testing, helps mitigate emerging risks.

Delivering quality products at a faster pace reduces the amount of time available to consider how to handle or treat each risk, so in DevOps, automated controls are utilized for a reduced risk of manual error. However, beyond automation, human judgment is needed in identifying, evaluating, and escalating risks and the effectiveness of risk treatment.

In DevOps, risk identification and treatment are proactively and automatically made visible to stakeholders to support decision-making, in alignment with organizational policies and procedures.

Example: If there is a requirement for a cloud provider to provide a specific level of security, the requirement cannot be changed without the customer's accepting the change in the level of risk.

When multiple organizations are collaborating on the production of a system, prudent risk management indicates a need to integrate their risk perspectives into a single view. Dependencies throughout the DevOps supply chain (the ecosystem) need to be recognized, which is why unmitigated vendor risks cannot

be accepted on behalf of the project or organization when delivery needs to be guaranteed. When vendors are not operating at the same level of quality, the non-achievement of any dependency can affect team velocity, a vulnerability that needs risk management through the Acquisition (6.1.1) and Supply (6.1.2) processes.

In DevOps, traditional top-down control is less evident due to the empowerment of teams. In addition, techniques such as continuous test and monitoring, vulnerability scanning, chaos and penetration testing, and monitoring deployed within a Risk Management framework can help balance the priority and amount of testing required to reduce risk and assure quality and security beyond minimum legal and regulatory requirements. Adhering to Risk Management frameworks and processes diligently and consistently is a way to reduce adverse effects on the organization, other stakeholders, and the environment.

Table 1 identifies aspects of risk management that typically need consideration (6.3.4.3.a)8)) in line with the organization's risk management process and the specific requirements of the project to enable success of the system stakeholders in the DevOps ecosystem.

NOTE—Quality attributes can be identified from the ISO/IEC 25010 [B7] product quality model. Process risks are described in ISO/IEC TR 33015:2019 [B13].

Table 1—DevOps Risk Management considerations

Consideration	Example
Difficulty of meeting required SLO or acceptable level of the product's quality attributes	Requirement for reliability and 24-hour availability of a power station versus a support desk that only needs to be online 12 hours/day.
Procedures to alert stakeholders when the risk is crossing a threshold and now is out of bounds	Informing individuals or a population via text message when a risk occurs that could endanger human life or critical infrastructure
Communication practices that notify relevant stakeholders about risk treatments	Notice on a dashboard or tool that employees are overseeing, so that risks will not be overlooked.
The human factors involved with being able to process risk alert information	Appropriate speed and frequency at which alerts are displayed when automated risk monitoring indicates that risk thresholds are about to be exceeded.
Risk appetite/or tolerance for social, environmental, or financial risks	When flooding destroys communities designed with planning software that ignores the potential consequences of building in a flood plain.
How policies drive escalation processes and appropriate risk response in a feedback loop	Failure to escalate, misalignment of escalation processes across all stakeholders, or not taking escalations seriously. Engineers informed Space Shuttle Columbia's management of O-ring issue affecting launch at low temperatures, but the risks were not considered at higher management levels.
How decision-makers will manage escalations (including release decisions) for risks that are approaching thresholds for a program of work, including how the escalations will be managed moving forward. Escalation thresholds can change, the resources responding to the escalation can change, and the response to the escalation can change.	A platform provider ignores employee or customer feedback on following their own policies until the bottom line is damaged or loss of life occurs. Risk process changes could require the provider to follow internal procedures using automation to remove human error and/or bias.
How resources are prepared for high performance	Assurance that the resources implementing all elements of risk planning are suitable for the rapid rate of change that occurs in DevOps. An organization using manual processes buys additional computers to improve product delivery velocity. An organization using automated processes uses the cloud to increase load support automatically. Assurance that skilled human resources are available to automate the risk identification and subsequent processes, which can include consideration of training, capability, competency, availability, authority level, and skill level requirements. Assurance that system resources, such as environments, infrastructure, and tools are available to automate the risk identification, monitoring, and treatment processes. Consideration of the depth of skills and knowledge required in management, project team members, and risk owners to manage risk activities according to quality objectives. Skills can be both soft and technical, as described in 6.2.6.3.

Table 1—DevOps Risk Management considerations *(continued)*

Consideration	Example
Facilitation of information-driven decisions, whether manual or automatic, by expediting and prioritizing upstream and downstream process or product defect estimates	Defect estimates include defect cost and quantity. SLAs that facilitate data-driven decision-making processes, based on risk thresholds and tolerance levels. These processes can be automated and integrated into the CI/CD pipelines in pre-production systems, as well as into live production monitoring systems.
QA audit of automated risk treatments and management of corrective actions	Processes to be audited include: Requirements engineering to identify which risks can be traced back to work products (e.g., task, story, requirement) Verification and validation to confirm that applied treatments have successfully controlled or mitigated each risk Transition to conform that bugs and other defects have been fixed and risks mitigated prior to approval for release
Degree of automation of risk management, unless there is evidence that the risk process cannot or should not be automated	Whether and how automatically identified risks need to be made visible, e.g., with a dashboard. Whether and which records need to be kept of each risk, alert, and treatment occurring, for what purpose (such as handling risk within the pipeline and continuous improvement), and how long the records are kept. Whether the issue/case management system and the CI/CD environment are integrated. Identification of processes and activities at risk, including (individually or in combination): architecture, content management, data warehouse/business intelligence (BI)/extract, transform, load (ETL), data quality and integrity, integration/APIs, metadata, operations, reference and master data, security, privacy, segregation of duties, testing, and quality practices.
Development and implementation of automated measurement practices that create actionable alerts, prior to risk thresholds being exceeded and any actions being taken	Measurements of lead time, deployment frequency, deployment time, mean time to detection (MTTD), number of employees relative to financial risk, number of non-compliant change requests.
Visibility over external supply chain controls	Decisions differentiating between different suppliers based on their capabilities and risk profiles.

6.3.4.2 Outcomes

As a result of the successful implementation of the Risk Management process, achievement of the following outcomes shall be demonstrable:

a) A structured and comprehensive strategy for risk management is defined that accommodates the dynamic DevOps environment.

b) The risk management approach is integrated into the DevOps deployment pipeline and other life cycle processes, policies, and procedures.

c) Automated monitoring practices are established to determine when risk thresholds are going to be exceeded, with each risk made visible.

d) Continuous compliance and continuous risk management result in acceptable risk related to DevOps for the stakeholders.

6.3.4.3 Activities and tasks

The organization shall implement the following activities and tasks in accordance with applicable organizational policies and procedures for DevOps with respect to the Risk Management process:

a) **Establish a risk management strategy or approach that integrates the dynamic DevOps environment into the organization's risk management framework**. This activity consists of the following tasks:

1) Establish the organization's level of risk tolerance or risk appetite.

2) Develop an organizational risk profile, based on individual risk profiles and including social, environmental, and financial risks.

3) Identify the roles and responsibilities required in establishing the automated risk management framework, including automation and manual stakeholder interventions requiring a human in the loop.

4) Establish policies to alert stakeholders regarding the following:

i) Risks that exceed risk thresholds

ii) Responsibilities for managing different risk types and priorities

iii) Effectiveness of risk treatments

NOTE—Human factors engineering is involved in determining the speed and frequency at which alerts are displayed and announced.

5) Reconcile project and product objectives of affected stakeholders with organizational risk policies.

6) Establish policies and automated systems to enable risk detection, decision-making at risk thresholds, risk response, and escalation as the probability and impact of risks change.

Example: The risk of international travel for face-to-face work with deployed teams is low at the start of a project, but increases exponentially over time, due to a pandemic virus outbreak.

7) Apply the risk framework consistently across project and system boundaries to prevent organizational discrepancies and supply chain disruptions.

NOTE 1— System of systems (SoS) risks (arising from internal or external system silos) are often hidden inputs to the risks for an organization, project, or system of interest.

NOTE 2— The automated risk identification process or relationships that cross internal and/or external project boundaries can mask emerging risks.

8) Expedite continuous governance over risk-based decisions that aligns with the environment, the people and their skills, as well as the capability to evaluate and manage dynamic risks.

9) Obtain stakeholder approval of the risk management approach.

10) Activate the organization's risk management process within and across system, project, program, or organization boundaries.

b) **Identify and evaluate risks.** This activity consists of the following tasks:

1) Identify risks from the following sources and their risk owners:

i) Release decisions

ii) SLA, leadership, and oversight commitment to customer quality, stakeholder engagement, and pipeline velocity

iii) Adequacy of process capability to meet DevOps process objectives

NOTE—Process capability includes the DevOps environment's ability to accommodate acceleration, varying skill sets and experience levels, and measurement of outcomes with respect to established risk tolerances.

iv) Design or operational processes and procedures, such as architecture, content management, data warehouse BI and ETL, data quality and integrity, integration and APIs, metadata, reference and master data, security, privacy, segregation of duties, testing, and quality practices

v) Emerging risks from causes such as effects of programmatic changes and newly automated procedures on current projects

Example: When budgets change without end-to-end consideration of the impacts to products and services being offered within that budget scope, process failures and breakdowns can occur, leading to losses in customer satisfaction and organizational reputation.

vi) Technical deficiencies such as security violations, unapproved changes, or data quality

vii) Interactions and relationships between risks, as products/systems are iteratively built and established

viii) Management of opportunities and risks, consistent with the velocity of the pipeline

ix) Management of incidents, issues, and problems

2) Assess the likelihood of risks.

3) Identify the potential consequences (impacts) of untreated (or partly treated) risks on the organization, stakeholders, the environment, and the public.

NOTE—Impact includes the cost of identifying, managing, treating or ignoring risk, and remediation costs when unmitigated risks eventuate as defects or incidents. Risk mediation can introduce a range of new risks that are more impactful and costly than the original risk.

Example: An engineer decides not to raise a risk that human life could be endangered, due to the additional allocation of budget required to fully treat the risk. By not escalating such risks, an organization or project can incur an unacceptable outcome that puts the organization itself at risk of permanent and catastrophic damage.

4) Establish the relative priority of treatment for risks, in consideration of capability and response criteria and consistent with the organization's risk profile and regulatory compliance requirements.

Example: Treating risk may delay shipment of a release, which could ultimately be more cost-effective than early release of a system with defects. An organization can deem that any risk that could endanger human life shall be treated to a sufficient degree as to prevent any reasonable possibility of the risk materializing.

c) **Apply and monitor risk treatments.** This activity consists of the following tasks:

1) Engage in continual risk monitoring, analysis, response, and escalating notifications that are automated and integrated into the CI/CD pipelines, wherever possible.

2) Reduce risk by applying controls, such as the following:

i) Automation of integration and release activities, which reduces manual error and increases the requirements and pace at which teams can identify and treat risk

ii) Automated measurement tools that create actionable alerts before risk thresholds are exceeded

Example: Types of risk measurements could include lead time, deployment frequency, deployment time, mean time to detection, number of employees relative to financial risk, number of non-compliant change requests.

i) Involvement of stakeholders (e.g., developers and operations staff), which induces buy-in and improves knowledge sharing and timely communications

ii) Embedded systems views and thinking, which preserve the holistic nature of the system being delivered

iii) Fast feedback loops that point out where errors have occurred, enabling faster resolution, including transparent repositories of information through logging and subsequent analysis of the logs

iv) Deployment of small batch sizes that introduce small, controlled, well-tested changes

v) Releasing to a smaller number of users per release and measuring the success of each release, to reduce the impact of potential failures on users and the organization

vi) Automated measurement of testability and testing outcomes that determine the level of risk being introduced with each batch

vii) ·Re-teaming when teams become ineffective at achieving project goals

3) Record the results of risk-taking and innovation using the Knowledge Management process or risk repository.

NOTE—Risk records include alerts, treatments, and related incidents and problems.

4) Accommodate automated risk detection and resolution within the organizational risk management framework in support of continuous improvement and higher levels of automation.

d) Continuously identify and communicate risks, risk alerts, risk treatments, and results and business value achievements visible to relevant stakeholders.

Example: Graphical representations (e.g., information radiators), large and frequently updated displays of project information that is continually visible to the risk owners, project team, and other stakeholders

e) Audit risk management and expedite continual corrective actions and process improvements, enabling risk identification, treatment, alerting, and results to be more readily applied.

NOTE—Audits include outlier areas, such as where manual activities or complex and novel automation has been implemented. Corrective actions and continuous improvements take into account whether the risk response is appropriate for the relevant stakeholders, including the effect of the decisions resulting from automation.

6.3.5 Configuration Management process

6.3.5.1 Purpose of the process

The purpose of Configuration Management is to manage and control system elements and configurations over the life cycle. Configuration Management (CM) also manages consistency between a product and its associated configuration definition. (ISO/IEC/IEEE 12207:2017, 6.3.5.1)

Configuration Management (CM) is essential throughout the product's life cycle, from inception (identification of the need for the product) through its end-of-life disposal. The intensity of CM activity, however, varies according to different technical processes being invoked during the life cycle. Configuration Management in this document includes both the IT assets and the activities of organizational change management.

NOTE—IEEE Std 828 provides detail on CM activities, tasks, and principles.

Configuration Management traditionally embraces the four core practices of configuration identification, status accounting, change control, and configuration audit (e.g., physical and functional). For software, these practices are largely focused on source code management—build, release, and deployment engineering—along with status accounting as the basis for change control, and environment and application life cycle management (ALM). ALM tools are commonly used for status accounting and effectively manage the software and systems configurations for DevOps. For DevOps of software, the "physical" configuration audit is a verification that software components have only been changed as authorized. Verification of the integrity of software items uses encryption hashes (e.g., MAC SHA1, MD5) as part of the physical configuration audit.

Many of the best practices found in CM have evolved and taken on greater focus in DevOps. In DevOps, CM supports the need to identify, select, and control changes to configuration items including systems and applications build, package, and deployment (as continuous delivery [CD]). These efforts require the strategic use of source code management systems, automated build, package, and deployment, as well as baseline verification. CM tracking of software and system artifacts should be part of integrated and automated management that includes knowledge, risks, security requirements and concerns, interfaces, compliance, and other requirements that affect configuration items.

There is an increasing demand in the marketplace and by regulators that organizations achieve better outcomes with better managed risk. These expectations include not exceeding risk thresholds without prior agreement. When processes or practices change within an organization, such as when practices are transitioned to DevOps, risk and cost can also be expected to change. Each change requires change management, so that the result is a positive improvement. Change management includes culture, tools, processes, and practices to achieve the level of developmental capability required to achieve DevOps objectives.

Managing these changes can involve training, mentoring, and leadership coaching. Additional costs can include hiring and onboarding experts to lead the change. During a transition to DevOps, portfolio risk management can be required so that existing teams and projects are not damaged by redeployment to new projects, leaving other projects vulnerable to failure. Team velocity can be impacted during the change process, so to adapt, plans and schedules may need to be adjusted. Costs can also include lifting organizational process capability to a higher level, so that DevOps practices can be embedded throughout the organization and implemented successfully on a team basis.

6.3.5.2 Outcomes

As a result of the successful implementation of the Configuration Management process, achievement of the following outcomes shall be demonstrable:

a) The chain of evidence is verifiable from source code baselines through persisted derived objects to verifiable baselines.

b) The CM framework enables confirmation that the intended configuration items have been deployed and identifies any unauthorized changes (due to inadvertent error or malicious intent) to controlled environments.

c) Managed changes are accepted and applied by the affected stakeholders.

6.3.5.3 Activities and tasks

The organization shall implement the following activities and tasks in accordance with applicable organization policies and procedures for DevOps with respect to the Configuration Management process:

a) **Identify configurations and configuration items.** This activity consists of the following tasks:

   1) Identify and control configuration assets for critical systems in an automated asset management system.

      NOTE—Real time systems or up-to-date records are preferable.

   2) Track the evolution of CIs created during the software and systems life cycle.

b) Manage the configuration of software being continuously integrated. This activity consists of the following tasks:

   1) Build code from verifiable baselines.

   2) Embed immutable version IDs into configuration items.

c) **Manage configuration changes.** This activity consists of the following tasks:

   1) Plan configuration changes needed to mitigate risks in consideration of regulatory requirements.

   2) Establish management of information security for control of operational, development, and test assets.

   3) Establish mechanisms for routine approvals of low-risk changes to enable continuous delivery.

   4) Notify affected stakeholders of configuration changes.

   5) Maintain current and accurate records of the system configuration.

   6) Create a list of configuration items changed (manifest or bill of materials), protected from unauthorized change with checksum hashes.

d) **Perform CM auditing.** This activity consists of the following tasks:

   1) Conduct functional configuration audits and communicate results.

   2) Audit the system configuration periodically to avoid configuration drift.

   3) Apply automated procedures to verify the integrity of baselines (e.g., physical configuration audit).

6.3.6 Information Management process

6.3.6.1 Purpose of the process

The purpose of the Information Management process is to generate, obtain, confirm, transform, retain, retrieve, disseminate and dispose of information, to designated stakeholders.

Information management plans, executes, and controls the provision of information to designated stakeholders that is unambiguous, complete, verifiable, consistent, modifiable, traceable, and presentable. Information includes technical, project, organizational, agreement, and user information. Information is often derived from data records of the organization, system, process, or project. (ISO/IEC/IEEE 12207:2017, 6.3.6.1)

Every life cycle process generates data, and information items are outputs (products) of every life process. The Information Management process establishes and performs the procedures for handling information products throughout the DevOps pipeline. Information Management is distinguished from Knowledge Management and Configuration Management by content. Organizations generate information to meet the needs and requirements of stakeholders. Information is based on data records and context.

NOTE—ISO/IEC/IEEE 15289 [B17] provides details on the content of information items and records related to life cycle processes, with a general procedure for preparing information items.

Information gathered by the management process is traceable to business goals. Information has both data and context available for reference and is accurate, clear, and consistent.

Example: Records management handles how the logs of an application's transactions are kept for two years and stored in an encrypted and stable form and have a specific format and definition of content.

With the larger number of stakeholders (producers and consumers), higher levels of automation in both collection and consumption of information is required. Automation is needed for the higher demand of customization, delegation, ease of use, and change of these services. The goal is to enable producers and consumers the capability to rapidly and easily gather, store, transform, and retrieve the information through a predetermined retention period.

Information should be provided in a timely manner, enabling immediate action in emergency situations, but if time permits, presented in context and with enough lead time for review and decision-making in consultation with stakeholders.

6.3.6.2 Outcomes

As a result of successful Information Management process, achievement of the following outcomes shall be demonstrable:

a) Stakeholders (information producers and consumers) are included in a transparent and collaborative manner and provide continuous feedback and improvement to the processes and systems.

b) The timeframes of production and consumption of information are appropriate to producer and consumer.

NOTE—The collection and reporting rhythm can be from real time to periodic.

c) Tools, services, and systems used to implement information management processes are integrated into relevant pipelines in the ALM processes.

6.3.6.3 Activities and tasks

The organization shall implement the following activities and tasks in accordance with applicable organization policies and procedures for DevOps with respect to the Information Management process:

a) Identify which stakeholders should have access to which data, considering security and privacy.

NOTE—Producers and consumers can be allowed to self select according to the appropriate authentication and authorization of access to the information.

b) Communicate activities and results to stakeholders so that the business or mission value is evident.

c) Collect the information inventory in an automated mechanism, including such attributes as:

— The information producers and consumers

— References to related contextual descriptions

— The sensitivity and privacy restrictions on the information

— The retention and removal of the information

d) Select the tools, services, and enabling systems used in information management activities.

NOTE—The information, tools, services, and systems relating to the information management processes are secured, retained, and disposed appropriately to the demands of the sensitivity, privacy, and laws and regulations applicable to the information and processors. Implement and establish communication channels (distributed team communication) to enable connections and trust relationships for both on-premise and remote stakeholders.

6.3.7 Measurement process

6.3.7.1 Purpose of the process

The purpose of the Measurement process is to collect, analyze, and report objective data and information to support effective management and demonstrate the quality of the products, services, and processes. (ISO/IEC/IEEE 12207:2017, 6.3.7.1)

DevOps brings quantitative visibility to aspects of the organization value streams at levels such as project, program, portfolio, and organization. Measurements can relate to projects, products, or processes. Measurement impacts roles in the organization, including developer, testing, operator, security officer, and management. Any role should be able to readily implement and refine measurements to support problem analysis and evidence of work progress and process improvement.

Automated measurements are quantifiable and shall be related to technical or management decisions. Qualitative (scalar) measurements can be used when human judgement is important. When possible, automation of measurement is preferred. In DevOps, often the amount of information ingested and data collection rates cannot be managed without automation. DevOps prioritizes ease of adding generation points (sources) created by producers and ingestion points (sinks) by consumers, while providing customizable filters and transformations in order to be of immediate use to the various consumers, through methods such as ETL, subscription streams, dashboards, and control systems.

Measuring technical debt and unplanned work (waste) can contribute to improvements such as:

— Realistic error budgets

— Left-shift (earlier) alignment between sustainable engineering velocity and business goals and expectations

— Improvements to delivery process to meet business goals

— Adjusting business goals based on current capacity to deliver (engineering reality)

DevOps creates dynamic visibility for organizational levels and areas for many of the artifacts that have been traditionally static. For example, architectures, rather than simply documented statically, can be continuously monitored and measured for the amount of change in their characteristics. Measurements shall be refined with at least the same or greater rate than the change in the application and its use.

Measures depend on a defined domain model or contextual framework understanding so that interpretation is consistent and appropriate. Without an accurate record of the context in which data was gathered, decisions cannot be reliably drawn from it.

Measurement is aligned with other continuous integration and deployment activities. The DevOps concept of “all infrastructure as code” applies to measurement, and therefore measurement is performed as part of the CI/CD value stream without entailing manual collection, aggregation, and reporting.

Example: The following are typical measures for availability and performance that can be useful in DevOps applications and services:

- Deployment frequency
- Change lead time and volume
- Change failure rate
- Defect rate
- Mean time to recovery (MTTR)
- Mean time to detection (MTTD)
- Issue volume (number of issues) and resolution time
- Time to approval
- Time to patch vulnerabilities
- Logging availability
- Retention control compliance
- Software assurance requirements findings count
- Application traffic
- Attack vector details (IP, stack trace, time, rate of attack)
- Resource utilization

NOTE—ISO/IEC/IEEE 15939:2017 [B18] provides a common process and framework for measurement of systems and software projects and products across the life cycle.

6.3.7.2 Outcomes

As a result of the successful implementation of the Measurement process, achievement of the following outcomes shall be demonstrable:

a) Measures related to the application products, processes, and portfolios of the organization are clearly defined and tied to an information need (question) they answer, a goal they serve for one or more stakeholders, and a commonly held and documented contextual framework or domain model.

   NOTE—The domain model provides a common reference so that relevant stakeholders can consider the measures with the same underlying understanding of the context, concepts, roles, data types, individuals, and rules.

b) Measurements are analyzed and cross-correlated across the whole project, application, program, portfolio, or organization to meet information needs and support decision-making.

c) Data results are delivered in the timeframes of need, up to, and including near real time.

d) Security and privacy controls on measurement data are imposed only where needed, enabling broad visibility and avoiding artificial visibility constraints (i.e., departmental and role silos).

6.3.7.3 Activities and tasks

The organization shall implement the following activities and tasks in accordance with applicable organization policies and procedures for DevOps with respect to the Measurement process:

a) Define the measurement strategy.

1) Relate the prioritized goals of the organization, program, portfolio, project, product, or application or work to the design of measures.

   NOTE—Un-prioritized goals without a clear contextual framework or domain model reference often lead to inappropriate, uncorrelated, or wasted measurement data.

2) Determine the questions to be answered to meet the goals of the organization, project, product, application, or work at hand.

3) Define and review how decisions are made based on measurements, analytics, and feedback controls.

4) Identify measures which produce actionable information for people and systems.

5) Define the measures to be collected, the purpose of the collection (what value each measure brings), and the tasks associated with collecting them.

6) Refine measures against the following characteristics:

   i) Quantifiable: If time-series based, consider appropriate sample rates.

   ii) Usable or actionable: Remove if not in service to a stakeholder or goal.

   iii) Measurable to the required level of precision.

   iv) Validated or verifiable.

b) Collect and analyze measurements.

1) Implement project plans to enable the automated collection, communication, and analysis of measurements supporting stakeholder-defined objectives and measures.

2) Apply automation to ingest, manage, and output accurate and available information.

3) Document procedures for measurement collection methods and tests of data validity.

c) Analyze and report information derived from measurements.

1) Analyze and interpret collected measurements and results.

2) Provide measurements aligned with the needs of the human and system consumers through appropriate interface mechanisms (e.g., dashboards or API).

3) Evaluate the cost of measurement collection and communication against the priority of the goal which the measure serves.

4) Continuously refine and trim measures and measurement activities to meet the changing needs of the organization in a timely manner and reduce waste and information overload.

d) Establish automated systems and tools for measurement.

1) Align Measurement to Security and Risk Management processes, so that the collection framework includes secured storage, transmission, and access.

2) Determine if measures are already available, or need additional instrumentation for collection, analysis, and reporting.

3) Select or develop a framework and tools to allow coordination of measurement collection for analysis, reporting, and control.

4) Establish a centralized data store and data retention policies for measurements that support automated ad-hoc or periodic comparisons and cross-correlation.

5) Support the integrity and quality of measurement data being collected.

6) Design, build, or configure instrumentation to collect metrics within the context of the application development processes, enabling measurement-driven design.

7) Support change management in the application life cycle through measurement.

8) Operationalize and maintain the collection framework infrastructure to accommodate needs such as reliability, availability, and capacity of the collection of metrics, analysis, and resulting report data.

9) Apply appropriate monitoring of the measurement systems (i.e., who watches the watcher).

6.3.8 Quality Assurance process

6.3.8.1 Purpose of the process

The purpose of the Quality Assurance process is to help ensure the effective application of the organization's Quality Management process to the project.

Quality Assurance focuses on providing confidence that quality requirements will be fulfilled. Proactive analysis of the project life cycle processes and outputs is performed to assure that the product being produced will be of the desired quality and that organization and project policies and procedures are followed. (ISO/IEC/IEEE 12207:2017, 6.3.8.1)

The Quality Assurance (QA) process assures the effective application of the organization's Quality Management (QM) process to the project. The QA process shall be put in place to ensure that these quality control activities are being completed in a continuous manner and that they are producing meaningful feedback on quality. The QA process shall map QM program measures to project-specific implementation plans as a means of identifying gaps in meeting requirements. The project plan should have a QA component, harmonizing the project's quality policies and strategies with the organization risk policies.

QAs not the sole responsibility of a single organizational unit, but rather, is driven by the QM and QA processes when mapped into the context of project work. QA assures activities are done. In DevOps, QA focuses on the following concerns:

a) Achieving the customer's objectives, and in turn, business or organizational goals and values.

b) Evaluating process outcomes continuously.

c) Speeding up delivery.

d) Providing evidence to make informed decisions and achieve customer satisfaction.

QA applies to data, process, and product quality. QA activities in DevOps serve as oversight to assure that the requirements are being met as the deliverables flow through automated pipelines. QA provides confidence based on objective evidence that the prescribed processes have been followed with adequate competence, and according to approved plans, to meet their expected outcomes. QA includes monitoring the achievement of project-specific outcomes using automated tools which collect evidence (e.g., KPIs, indicators, failures) that can be used to measure goal achievement and refine the goals. QA is not equivalent to testing, since testing is an activity associated with processes, such as verification and validation, performed by various contributors through the system delivery life cycle.

QA as a mindset applies to Technical processes (6.4) where incomplete or malformed outcomes of upstream work negatively impact the activities in downstream work and can rapidly decrease overall velocity. In DevOps, QA is started early in the system life cycle. Shifting quality to the left means adopting a proactive approach so that that even the earliest activities in the process are ready to execute properly. This left-shift is performed to design and engage the practices necessary to assure system quality throughout the entire life cycle. QA surfaces defects early in development processes to reduce rework and errors during transition and production operations. The efficiencies that DevOps practices offers to QA activities can enable faster time-to-market and improved product quality compared to more traditional methods of delivery.

QA activities shall be adapted for automated delivery pipeline requirements, such as early levels of coverage (e.g., unit, integration, and API), timeliness of feedback to contributors, and analysis of results to discover actionable improvements in quality. In this way, QA is a continuous process to elicit feedback that is relevant within development and operational cycles, not a separate stage in the life cycle.

QA verifies that practices that are geared toward achieving business aims, lead to a reduction in time-to-market, and are executed according to approved protocols. QA shall determine if a product is meeting specified acceptance criteria for a set of requirements (business or technical) through a review and examination of the way the process itself was executed, its artifacts, plans, and timelines. QA has a key role in the implementation process, aiding feedback communication (e.g., what happened during handoffs between contributors, and the impact of new requirements, new dependencies, and new risks). Therefore, the QA process incorporated in DevOps shall oversee continuous integration, continuous testing, and continuous feedback to developers.

QA activities include oversight of verifying (testing) and validating (reviewing with relevant stakeholders). Although QA is often confused with testing because of the perceived overlap with verification and validation activities, QA and testing are distinctly different activities. QA confirms that testing activities, as well as other processes necessary to attain quality outputs and outcomes, are performed as approved in project planning. QA consolidates assurance that automated testing processes produce automated testing outcomes, including pass/fail results, SLO and SLI failures, identified security vulnerabilities, compliance status checks, release-dependent go/no-go determinations, and other indicators supporting automated build/deploy processes. QA should have the authority to stop a deployment due to compliance, risk, and security concerns.

In production, the QA activities should be performed non-obtrusively. To support this goal, the system design should include the collection of test data (telemetry points) in an automated test framework (test harness). The test harness supports monitoring of the system operations including the supporting infrastructure. This includes continuing to monitor quality during production. The results of the monitoring process shall be continuously evaluated. Situations that reflect anomalous behavior shall be flagged and when possible automatically addressed to promote the continuous, uninterrupted operation of the system.

A major focus of QA is identification of process irregularities which can lead to product and/or service defects as a result of deviating from the planned and approved process. QA uses automated monitoring and control (“sense and response”) mechanisms in DevOps. Defects are classified according to their severity and outcomes. Automated quality gates and feedback loops help identify process defects and their severity, which in turn become pointers to product and service defects. These often exist as discrepancies in project and pipeline specific implementations of system verification, validation, monitoring, and telemetry processes that directly map to program-level key metrics such as SLOs. Through automated implementation, QA also seeks to identify coverage variances, critical information gaps, and unsatisfactory production service levels.

The QA process shall prove system readiness by assuring that automated validation and verification (V&V) measurement processes for functional, operational, and acceptance requirements are operational in a timely manner. The net benefit of this proactive behavior is that DevOps is not creating future unplanned work or technical debt that inevitably reduce velocity.

Examples: Code reviews and coding standards improve the quality of code as it is developed. The implementation of continuous testing rapidly increases test execution cycles, enabling improved quality and shorter release cycles. Implementing continuous test suites in areas such as performance, security, and accessibility directly reduces business risk via the early reduction in non-functional defects. Establishing dedicated teams with embedded testers also enables earlier uplift in product quality, leading to reduced time-to-market and further risk reduction. Arranging for development and operations staff to participate in early risk assessments enables a more complete understanding of system requirements, improves the completeness and effectiveness of subsequent testing, and leads to an improvement in product quality.

QA also assures that cross-functional feedback between development and operations staff enhances each DevOps delivery team's ability to prevent future defects, further speeding up release cycles. In each case, evidence-based optimization to QA processes are sought via more efficient and effective feedback loops.

QA data capture in DevOps typically involves new sets of process and quality data to determine if automated delivery pipeline requirements are being met and to provide oversight over the clarity of remediation actions. The QA process may also apply new methods of automated analysis so that the "sense" can be paired to the right "response" for actionable improvements in quality. New sets of data and methods of analysis result in continuous activities that are relevant to both development and operations. Setup, execution, and maintenance work to support QA shall be included in time and resources cost estimates.

QA processes shall align process metrics and measurements over functional, non-functional, and acceptance criteria to goals as evidence of project progress. Automated process checks, tests, and process monitoring should be used to measure service level indicators (SLI) which enable the QA team and other stakeholders to assess the readiness of deliverables as defined by service level objectives (SLO) and evidence of accuracy in the current quality measurement processes. Fulfillment of service level agreements (SLA) requires fulfillment of appropriate SLO through evidence in SLI and DevOps processes. QA tasks include continuous monitoring to provide real time feedback about the quality of deliverables.

6.3.8.2 Outcomes

As a result of the successful implementation of the QA process, achievement of the following outcomes shall be demonstrable:

a) Assessments are performed of organization, program, project, and system implementation of DevOps principles and practices.

b) QA guidance and assistance are provided across all portfolios and teams.

c) Evidence is provided that confirms whether or not deliverables meet business requirements, are made available to relevant stakeholders, and are consumable by recipients involved in addressing issues identified.

d) Continuous analysis of factors, trends, and information gaps contributing to quality-related issues are provided to QM and other relevant stakeholders.

6.3.8.3 Activities and tasks

The organization shall implement the following activities and tasks in accordance with applicable organization policies and procedures for DevOps with respect to the QA process:

a) **Establish and monitor the use of processes, automated tools, and procedures to build in quality and avoid quality issues.** This activity consists of the following tasks:

   1) Devise and implement automated prevention methods with developers and related stakeholders.

   2) Assure that relevant activities, such as planning, resource allocation, verification, validation, results delivery, compliance, measurement, and monitoring are functioning acceptably for deliveries to stakeholders.

   3) Confirm traceability of changes and requirements across automated pipelines.

   4) Prioritize implementation of feedback mechanisms on process performance, based on project-specific goals and objectives, and aligned with organizational goals and risks.

b) **Conduct technical reviews and walk-throughs, and monitor testing to identify anomalies and defects.** This activity consists of the following tasks:

1) Confirm that requirements match business needs and that relevant stakeholders understand and validate the context of features, changes, and expectations.

2) Assure the implementation and update of methods to improve the timeliness and completeness of test results, balancing risk coverage and latency of feedback accordingly.

3) Devise and implement automated methods of detecting anomalous behaviors within the system (application and infrastructure).

4) Analyze results of automated testing.

5) Conduct technical reviews and walk-throughs of test plans and models to improve defect visibility, resolution, and prevention.

6) Apply standard service level indicators (SLI) in accordance with project measurement and QM processes, including project-specific metrics when necessary.

7) Provide real time visibility over critical quality issues that affect service level agreements (SLA) through continuous SLO monitoring.

c) **Analyze, track, and report timely resolution of issues and problems.** This activity consists of the following tasks:

1) Trace defect reproducibility to their sources (in hermetic and ephemeral environments).

NOTE—Corrective actions consider the root cause and potential technical solutions on how to resolve gaps.

2) Document the context of contributing factors on identified defects and resolution steps (for example: postmortems, five why's, retrospectives, root cause analyses).

3) Confirm that when reviews and reconciliation activities identify a gap in fulfilling requirements, a corrective action is created, assigned, recorded, completed, and managed.

4) Confirm that applied controls and mitigations have successfully treated each risk through verification and validation.

5) Confirm that defects have been removed or acceptably resolved prior to approval for release being granted.

d) **Record and provide evidence of QA activities.** This activity consists of the following tasks:

1) Maintain evidence that verification, validation, and acceptance processes have been executed as planned and approved.

2) Align SLI-based evidence with service level objectives (SLO) so that stakeholders can easily understand the impact of service changes on deliverables' quality.

3) Drive cross-functional feedback aligned to key objectives defined by the QM process to produce evidence of system readiness, conformance, and improvements.

e) **Continuously improve QA activities.** This activity consists of the following tasks:

1) Reintegrate learning and improvements resulting from QA processes into future project plans.

2) Recommend adjustments to QA and QM policies and processes to adequately address identified risk areas and process gaps.

3) Incorporate new factors, tools, and methods into QA planning activities.

6.4 Technical processes

The Technical processes are used to define the requirements for a software system, to transform the requirements into an effective product, to permit consistent reproduction of the product where necessary, to use the product to provide the required services, to sustain the provision of those services, and to dispose of the product when it is retired from service.

The Technical processes define the activities that enable organization and project functions to optimize the benefits and reduce the risks that arise from technical decisions and actions. These activities enable software systems and services to possess the timeliness and availability, cost effectiveness, functionality, reliability, maintainability, producibility, usability, and other qualities required by acquiring and supplying organizations. They also enable products and services to conform to the expectations or legislated requirements of society, including health, safety, security, and environmental factors. (ISO/IEC/IEEE 12207:2017)

6.4.1 Business or Mission Analysis process

6.4.1.1 Purpose of the process

The purpose of the Business or Mission Analysis process is to define the business or mission problem or opportunity, characterize the solution space, and determine potential solution class(es) that could address a problem or take advantage of an opportunity. (ISO/IEC/IEEE 12207:2017, 6.4.1.1)

In DevOps, the Business or Mission Analysis process emphasizes the use of automation in order to better address the needs of the business/mission by means of maintaining current and continuous alignment across relevant stakeholders when characterizing, defining, and maintaining a shared vision for the solution and removing sources of defects and delays.

Engaging and collaborating with relevant stakeholders early in the process (i.e., left-shift) reduces risk by providing the decision makers with information required to make informed choices. Using automation, in alignment with organizational policies (especially security and privacy) to facilitate collaboration, communication, transparency, access to up-to-date reports, and the removal of sources of process delays is another significant way DevOps reduces risk.

6.4.1.2 Outcomes

As a result of the successful implementation of the Business or Mission Analysis process, achievement of the following outcomes shall be demonstrable:

a) The problem or opportunity and the solution space are defined collaboratively and communicated in a timely manner through automation.

b) The solution space is characterized with an emphasis on quality attributes, especially resilience, security, and privacy.

c) Solution space operational concepts required to establish, utilize, maintain, and improve the solution apply automation, continuous integration/continuous deployment, and continuous monitoring.

d) Traceability of business or mission problems and opportunities and the preferred alternative solution classes are established.

6.4.1.3 Activities and tasks

The organization shall implement the following activities and tasks in accordance with applicable organization policies and procedures for DevOps with respect to the Business or Mission Analysis process:

a) **Prepare for Business or Mission Analysis.** This activity consists of the following tasks:

1) Define, document in a central repository (within privacy and security policy constraints), communicate, maintain, and enhance the business or mission analysis strategy.

2) Continuously review with relevant stakeholders identified problems and opportunities in the organization strategy with respect to desired organization goals or objectives.

3) Identify, document in a central repository (within privacy and security policy constraints), communicate, maintain, enhance, and plan for the necessary enabling systems or services and automations needed to support business or mission analysis.

4) Obtain or acquire access to the enabling systems or services to be used across the entire life cycle.

b) **Define the problem or opportunity space.** This activity consists of the following tasks:

1) Analyze stakeholder complaints (especially from customers and users), problems and opportunities in the context of relevant trade-offs and domain-specific factors.

2) Define, document in a central repository (within privacy and security policy constraints), communicate, maintain, and enhance the mission, business, or operational problem or opportunity.

c) **Characterize the solution space.** This activity consists of the following tasks:

1) Define, document in a central repository (within privacy and security policy constraints), communicate, maintain, and enhance the operational concepts by means of automation.

2) Identify regulatory compliance constraints for the domain.

3) Identify, document in a central repository (within privacy and security policy constraints), communicate, maintain, and enhance candidate alternative solution classes that span the potential solution space.

d) **Evaluate alternative solution classes.** This activity consists of the following tasks:

1) Assess, document in a central repository (within privacy and security policy constraints), communicate, maintain, and enhance each alternative solution class.

2) Select, document in a central repository (within privacy and security policy constraints), communicate, maintain, and enhance the preferred alternative solution class(es).

e) **Manage the business or mission analysis.** This activity consists of the following tasks:

1) Maintain traceability of business or mission analysis in alignment with the organizational security and privacy policies.

2) Provide, document in a central repository (within privacy and security policy constraints), communicate, maintain, and enhance key artifacts and information items that have been selected for baselines in alignment with the organizational security and privacy policies.

3) Establish, employ, and through use, continually enhance organizational and project policies and procedures to automate disaster recovery and resilience tasks as feasible and relevant to improve availability and resilience and to fully recognize and treat risks.

6.4.2 Stakeholder Needs and Requirements Definition process

6.4.2.1 Purpose of the process

The purpose of the Stakeholder Needs and Requirements Definition process is to define the stakeholder requirements for a system that can provide the capabilities needed by users and other stakeholders in a defined environment.

It identifies stakeholders, or stakeholder classes, involved with the system throughout its life cycle, and their needs. It analyzes and transforms these needs into a common set of stakeholder requirements that express the intended interaction the system will have with its operational environment and that are the reference against which each resulting operational capability is validated. The stakeholder requirements are defined considering the context of the system-of-interest with the interoperating systems and enabling systems. (ISO/IEC/IEEE 12207:2017, 6.4.2.1)

DevOps emphasizes collaboration with stakeholders. The information considered under the scope of the management process and the range and quantity of stakeholders is often much broader than in traditional practice. In DevOps, the purpose of the Stakeholder Needs and Requirements Definition process is to:

a) Extend the stakeholder group more broadly beyond technical teams and into the business (including for example business units, legal, compliance, risk, privacy, and security teams).

b) Model the functional and non-functional requirements definition into the DevOps pipeline in a continuously automated and traceable way.

c) Allow any stakeholder to determine the posture of their functional or non-functional requirements against business risk using a self-service model.

Any changes to requirements captured within this process can lead to iterating through downstream stages of the life cycle, such as design, implementation, testing, and operation.

6.4.2.2 Outcomes

As a result of the successful implementation of the Stakeholder Needs and Requirements Definition process, achievement of the following outcomes shall be demonstrable:

a) Relevant stakeholders are identified.

b) A set of requirements that articulate stakeholder views of the system is continuously defined and refined.

c) Each stakeholder retains assurance of their requirements definition and acceptance criteria though integration of workflows.

d) Stakeholder requirements, including any changes to requirements, are agreed by relevant stakeholders.

6.4.2.3 Activities and tasks

The organization shall implement the following activities in accordance with applicable organization policies and procedures for DevOps with respect to the Stakeholder Needs and Requirements Definition process:

a) Identify stakeholders and their needs.

NOTE—The user is often the most important stakeholder to include in this process. Other stakeholders could include developers, testers, operations, business representatives, subject matter experts, HR, finance (that could review and approve budgets for system implementation), and any other department that will either utilize the target system or who have systems that integrate with the target system.

b) Develop an integrated view of disparate workflows so that each stakeholder retains assurance of requirements definition and acceptance criteria.

NOTE—Different stakeholders often hold different views of the requirements. For instance, a user might require a new system to provide single sign-on, while infrastructure might then require changes to their existing single sign-on solution, in order to accommodate that feature being included in the new system. This varying views on the requirements can be captured for a more complete model of the system, throughout system operations, usage, and support situations. In this sense, views can be functional, non-functional, or operational.

c) Analyze the views and risks of each requirement (story), both functional and non-functional.

Example: Stakeholder views can include user views, technical views of developers, operations, business and technical management, security, tester, or QA views.

d) Build a complete set of requirements by considering tradeoffs among stakeholder needs.

e) Obtain agreement on stakeholder requirements and acceptance criteria from relevant stakeholders.

NOTE—In dynamic, rapidly changing environments such as DevOps, requirements can change frequently, and this can impact the agreements made by stakeholders. Agreeing on requirements is typically iterative.

f) Continually and visibly prioritize, trim, communicate, and reconcile requirements across stakeholders.

g) Implement automated tools for machine-readable requirements artifacts.

6.4.3 System/Software Requirements Definition process

6.4.3.1 Purpose of the process

The purpose of the System/Software Requirements Definition process is to transform the stakeholder, user-oriented view of desired capabilities into a technical view of a solution that meets the operational needs of the user.

This process creates a set of measurable system requirements that specify, from the supplier's perspective, what characteristics, attributes, and functional and performance requirements the system is to possess, in order to satisfy stakeholder requirements. As far as constraints permit, the requirements should not imply any specific implementation. (ISO/IEC/IEEE 12207:2017, 6.4.3.1)

In a DevOps process, the requirements for a software system shall be based on validated stakeholder requirements that they have been fully captured and interpreted by the team.

Requirements should not be an unannotated sketch, or locked up in a system that requires manual copy of information in order to test against, or so ambiguous that they cannot be written down and understood. Requirements should be articulated in a format that works for humans and for automation.

Requirements should be traced and tracked in a manner that aligns with the overall software and systems development process. In DevOps, requirements implementation and tracing shall be able to keep up the rapid rate of change that can result from iterative development and emerging requirements. Requirements traceability benefits from, and can also be, a source of continuous feedback. Requirements should be traced from the beginning of the software and systems life cycle (i.e., left-shift). Organizations may trace requirements and associated test cases for audit and regulatory purposes.

Requirements definition includes requirements review for feasibility. The perspective of the ISO/IEC 25010 [B7] software/system product quality model can be applied, including considering whether a satisfactory level of quality characteristics is feasible with the proposed system requirements. Feasibility also includes consideration of the risk of implementing each requirement, both in terms of the system meeting each requirement (e.g., satisfying user needs) and in terms of the system not meeting each requirement (e.g., considering how the system might fail in production, for each ISO/IEC 25010 [B7] product quality characteristic). This consideration of feasibility and risk enhances requirement completeness and system robustness.

Since a large portion of production defects arise from errors in design and analysis, early review of requirements for clarity, consistency, and completeness can result in prevention or early detection and correction of errors. Requirements can be improved through analyst peer review, ambiguity tests, and prototyping.

When the infrastructure is the system of interest (SOI), the System/Software Requirements definition process shall include requirements pertinent to the scripting of infrastructure including networking, infrastructure as a service (IaaS), platform as a service (PaaS), or software as a service (SaaS) services (infrastructure as code).

The system requirements shall take into account the operational requirements which shall be put into place to enable the production and delivery of the final product.

6.4.3.2 Outcomes

As a result of the successful implementation of the Systems/Software Requirements Definition process, achievement of the following outcomes shall be demonstrable:

a) The defined systems/software requirements are feasible, measurable, testable, and consistent.

b) Requirements are traceable from stakeholder/user-oriented views of the system through the acceptance criteria, design, and implemented system.

NOTE—The aim of traceability is to identify requirements creep and requirement gaps.

c) Requirements include measurements relevant to DevOps, such as key service indicators and telemetry, service level objectives, and thresholds for feedback.

d) Emerging or changing requirements are managed, preferably by automated artifact generation.

6.4.3.3 Activities and tasks

The organization shall implement the following activities and tasks in accordance with applicable organization policies and procedures for DevOps with respect to the Software and System Requirements Definition process:

a) Capture product acceptance criteria in the system/software requirements.

NOTE—Product acceptance criteria are specified in the supplier agreement. They can include supporting manuals and materials that the supplier provides, and that stakeholders need to operate or support the system. Product acceptance criteria are supported by information items used by the stakeholders to determine whether the finished product is acceptable. Product acceptance criteria can include success and failure scenarios to support thorough testing.

Example: Product acceptance criteria can include supporting deliverables, meeting service level agreements, achievement of non-functional requirement levels, establishment of development and target operating environment requirements, user manuals, and transfer scripts to transition existing users onto the new system.

b) Automate the process for tracing requirements.

NOTE—Each technical requirement adequately supports a business need (stakeholder requirement).

Example 1: When a developer attempts to integrate code into the repository, if there is no case number (requirement reference) attached to the code change, then the code is rejected.

Example 2: For testing, when requirements are linked to automated or manual test cases, then if those requirements change, an automated alert can indicate that the test cases could require updating. Creating automated test cases and checking in/out test cases from the code repository also allows for tests to be rejected when they are not linked to requirements/case numbers. Running a test against that requirement can automatically indicate when a requirement is "done" when all tests pass.

c) Review requirements for feasibility, measurability, testability, consistency, and traceability.

NOTE—Each technical requirement is evaluated against relevant risk factors.

Example: Reviews can include peer reviews, walk-throughs, and technical inspections.

d) Establish iterative requirements management practices including managing emerging requirements, prioritizing requirement changes, and making them visible to relevant stakeholders.

NOTE 1— Emerging requirements can be defined when the system evolves over time. If requirements change, then the activities and tasks can be repeated so that artifacts still address the business problems.

NOTE 2— A product could be deemed to be compliant and meeting requirements. However, failure in the production environment could lead to a need to redefine performance (non-functional) requirements.

NOTE 3— Technical debt (unplanned deferred work usually requires refactoring and revision of requirements).

Example 1: If production monitoring detects issues (e.g., 2% of all requests fail) then this could drive the need for requirement change.

Example 2: Requirements might need to be updated when automated production monitoring detects failures or when there is a potential for not meeting service level agreements.

b) Communicate the software/system requirements, including any changes to requirements, to relevant stakeholders.

NOTE—Stakeholders could include those who use the system and those who build it. The set of technical artifacts that are defined need to articulate the system to people that are non-technical (for the people who need the system), as well as the technical staff (who build the system). They each have a different set of needs.

6.4.4 Architecture Definition process

6.4.4.1 Purpose of the process

The purpose of the Architecture Definition process is to generate system architecture alternatives, to select one or more alternative(s) that frame stakeholder concerns and meet system requirements, and to express this in a set of consistent views.

Iteration of the Architecture Definition process with the Business or Mission Analysis process, System/Software Requirements Definition process, Design Definition process, and Stakeholder Needs and Requirements Definition process is often employed so that there is a negotiated understanding of the problem to be solved and a satisfactory solution is identified. The results of the Architecture Definition process are widely used across the life cycle processes. Architecture definition may be applied at many levels of abstraction, highlighting the relevant detail that is necessary for the decisions at that level. (ISO/IEC/IEEE 12207:2017, 6.4.4.1)

DevOps principles and practices can improve the architecture definition process by enabling better communication and collaboration between stakeholders. Architecture definition is a flexible and scalable process throughout the life cycle endeavor. Due to changing needs, the architecture shall be extended for effective continuous delivery aligned with the architecture definition and business planning. For DevOps, the software and systems architecture shall support robust, secure, and scalable deployment. The architecture shall allow observability in support for monitoring system health and to drive scale up and scale down capabilities including the traceability of components between systems.

DevOps principles and practices mean that key stakeholders are involved and informed about architecture definition, knowledge is persisted and communicated appropriately, and effective feedback loops are established. The architecture and the related metadata should be available in consumable and accessible forms.

To manage risk, reference model architectures should be used. The compliance, legal, safety, security, privacy, and technical assumptions for the reference architecture shall be well understood. The changes to the reference architecture shall be made depending on context. The reference architecture library should be managed consistent with the organizational risk appetite and tradeoffs.

6.4.4.2 Outcomes

As a result of the successful implementation of the Architecture Definition process, achievement of the following outcomes shall be demonstrable:

a) Associated architectures are defined, described, and continuously assessed to support stakeholder needs.

b) The architecture applies the principles of security by design and privacy by design to code in the context of the operating organization and the users.

c) The architecture drives translation of legal requirements into technical data requirements for privacy.

d) Architecture artifacts and related metadata are available in consumable and accessible forms.

6.4.4.3 Activities and tasks

The organization shall implement the following activities and tasks in accordance with applicable organization policies and procedures for DevOps with respect to the Architecture Definition process:

a) **Define the architecture, including consideration of security, testing, operations, and maintenance.** This activity consists of the following tasks:

   1) Develop architecture alternatives that apply quality objectives with trade-offs and prioritization of choices.

   2) Coordinate and engage cross-functional subject matter expertise responsible for their respective domains to develop architectural definitions.

   3) Define a reference architecture that encompasses the views of the various stakeholders.

   4) Engage in continuous review and development of systems and applications reference architectures.

   5) Establish traceability of the model back to the requirements for the solution.

   6) Provide traceability from the architecture for testing and risk management, especially security and privacy risks.

7) Automate the generation and version control of architectural artifacts.

8) Communicate changes in the architecture to the relevant stakeholders.

b) **Incorporate security considerations in the architecture.** This activity consists of the following tasks:

1) Capture and disseminate related security information to apply to future architecture decisions.

2) Model and analyze the potential hazards to and from the architecture.

3) Enable and implement secure design criteria that have been reviewed by stakeholders for limitations, purpose, and relevance.

4) Evaluate the vulnerability of cybersecurity attack surfaces (both external and internal) to prioritize risk across the environments.

c) **Evaluate the architecture for correctness, consistency, and completeness.** This activity consists of the following tasks:

1) Establish risk-based governance of the architecture.

2) Build testing resources from the beginning of the architecture definition process, such as:

i) Reference test harnesses for system integration points

ii) Interface schemas and examples of data format (data schema)

iii) Mock-up or service virtualization artifacts in which dependencies are constrained by resource availabilities

2) Model verifiability (compliance with the model encompassing functional, reliability, performance, scalability, hazard, criticality, interface, security, and privacy aspects).

3) Model definitions which support automated static analysis and comparison between architectural reference, developed components, and system when in operation.

4) Determine whether model integrity levels are established for the stakeholder and system requirements, and detailed functions within that model.

5) Confirm that the selected architecture clearly defines the intended interaction of the system with its production operating environment.

6) Establish feedback loops from operations to verify and validate the security and reliability of the architecture.

7) Determine compliance of the selected architectures with standards, regulations, policies, and business rules.

8) Confirm that the system architecture satisfies the stakeholders needs.

6.4.5 Design Definition process

6.4.5.1 Purpose of the process

The purpose of the Design Definition process is to provide sufficient detailed data and information about the system and its elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture. (ISO/IEC/IEEE 12207:2017, 6.4.5.1)

A comprehensive approach to Architecture should enable effective patterns for design throughout the life cycle. DevOps principles and practices (e.g., left-shift) result in planning for QA, testing, and operations as the design is developed. The system shall be designed for automation of transitions, integration, and operations. The design process shall enable continuous changes in a timely manner throughout the life of the project.

Along with the fulfillment of functional requirements, the design process should account for the achievement of acceptable quality, the application of ethical values, and security and performance requirements. The design process for an infrastructure-based development project should include particular concern for safety. Generative code development for infrastructure services should include scheduled reviews for compliance with quality and ethical principles. Anything that is designed shall be verifiable and include relevant support (maintenance) considerations.

6.4.5.2 Outcomes

As a result of the successful implementation of the Design Definition process, achievement of the following outcomes shall be demonstrable:

a) The system design reduces the occurrence of errors.

b) Quality, ethical values, and security are designed into the product at the conceptual level when change is more economical and easier to effect.

c) Design decisions are traceable to the requirements and the implemented system.

6.4.5.3 Activities and tasks

The organization shall implement the following activities in accordance with applicable organization policies and procedures for DevOps with respect to the Design Definition process:

a) Conduct design reviews including ethical considerations and results of verification and validations.

b) Examine the dynamic interplay of execution in role-playing and simulations with the stakeholders.

c) Design systems for automated functions supporting DevOps processing.

d) Identify dependencies among systems and stakeholders affecting the design throughout the life cycle.

e) Agree on system interface specifications early in the process to facilitate parallel development.

f) Identify and record where and how information products from the system will be used.

g) Prioritize the design principles and characteristics.

h) Record the design definition with sufficient detailed data and information about the system and its elements to enable the implementation.

6.4.6 System Analysis process

6.4.6.1 Purpose of the process

The purpose of the System Analysis process is to provide a rigorous basis of data and information for technical understanding to aid decision-making across the life cycle.

The System Analysis process applies to the development of inputs needed for any technical assessment. It can provide confidence in the utility and integrity of system requirements, architecture, and design. System analysis covers a wide range of differing analytic functions, levels of complexity, and levels of rigor. It includes mathematical analysis, modelling, simulation, experimentation, and other techniques to analyze technical performance, system behavior, feasibility, affordability, critical quality characteristics, technical risks, life cycle costs, and to perform sensitivity analysis of the potential range of values for parameters across all life cycle stages. It is used for a wide range of analytical needs concerning operational concepts, determination of requirement values, resolution of requirements conflicts, assessment of alternative architectures or system elements, and evaluation of engineering strategies (integration, verification, validation, and maintenance). Formality and rigor of the analysis will depend on the criticality of the information need or work product supported, the amount of information/data available, the size of the project, and the schedule for the results. (ISO/IEC/IEEE 12207:2017, 6.4.6.1)

In DevOps, the design of networking, infrastructure, and computing systems and services shall be modeled and analyzed via their individual components and in their composite format as an end system in order to provide a rigorous basis of data and information to support technical understanding and decision-making throughout the design process. System Analysis during operations shall be supported through automation tools that monitor in real time critical system functions and take any necessary corrective actions to preserve system integrity, resiliency, and reliability; and to protect the user's experience. Systems analysis shall be considered from the beginning of the life cycle and include relevant stakeholders.

Automated analysis should be driven through the use of modeling, such as mathematical modeling, and other analysis techniques to anticipate changes that are needed in the system configuration. For example, modeling in this way aids in the early identification of potential competition for system resources (race conditions) that can be caused by automation feedback loops. Changes in the design for resilience can reduce the occurrence of these defects. Otherwise, once a potential race condition is identified, a more time-consuming impact on operations results if the system has to request intervention by a systems administrator.

Tools and models developed for system analysis purposes shall be committed to the code repository and procedures for their use should be documented.

An example of this type of support may be seen in a computation cluster where a node begins to fail in some way. The data and processing errors associated with the node are captured and analyzed by a monitoring tool that determines the failures are out of the bounds of tolerance. In response to the analysis, the monitoring tool triggers an automation script that gracefully shuts down the node that is experiencing the problem by removing it from the cluster and moving the processing and associated data to another node in the cluster.

Additionally, the automation process may configure a new node and introduce the node into the cluster. This may be done to expand the pool of resources as a way to relieve a resource constraint problem.

NOTE 1—Automation tools provide the opportunity to collect and sample data, analyze data, and react to data without the need for human intervention other than to set up the automation tool in the first place. Automation tools may be homegrown or commercially obtained. While each individual tool may only support one aspect of DevOps, automation is a fundamental concept in DevOps and tools are expected to be used throughout.

NOTE 2—Modeling provides a description of the system in such away as to reveal and anticipate operational characteristics and behavior under a variety of circumstances that may be too expensive or too time consuming to investigate otherwise.

6.4.6.2 Outcomes

As a result of the successful implementation of the Systems Analysis process, achievement of the following outcomes shall be demonstrable:

a) Results of applying analysis tools and models are available to both prove out architectural and design decisions and monitor operations.

b) Analysis tools and models are committed to a code repository and managed as any other system source code.

6.4.6.3 Activities and tasks

The organization shall implement the following activities in accordance with applicable organization policies and procedures for DevOps with respect to the Systems Analysis process:

a) Develop or obtain analysis tools and models to influence architectural and design decisions.

b) Analyze the results of monitoring operations and maintenance and to influence decisions, such as predicting failures and scheduling pre-emptive measures to avoid outages.

6.4.7 Implementation process

6.4.7.1 Purpose of the process

The purpose of the Implementation process is to realize a specified system element.

This process transforms requirements, architecture, and design, including interfaces, into actions that create a system element according to the practices of the selected implementation technology, using appropriate technical specialties or disciplines. This process results in a system element that satisfies specified system requirements (including allocated and derived requirements), architecture, and design. (ISO/IEC/IEEE 12207:2017, 6.4.7.1)

The Implementation process in DevOps is aligned to incremental development and continuous integration. The DevOps approach of transitioning between functional areas shall enable coordinated, cohesive parallel development efforts and feedback loops.

The Implementation process includes audit and regulatory sub-processes to audit and monitor feedback loops.

The Implementation process provides input and continuous refinement for version-controlled, timely information on the systems under construction, as well as systems that are part of the pipeline.

The Implementation process is performed iteratively along with the processes for integration, change control, risk management, and verification in preparation for continuous delivery (CD).