FINAL
DRAFT

INTERNATIONAL
STANDARD

**ISO/
IEC/IEEE
FDIS
32675**

ISO/IEC JTC **1**/SC **7**

Secretariat: **BIS**

Voting begins on:
2022-02-22

Voting terminates on:
2022-07-12

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

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

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 Software and Systems Engineering Standards Committee of the IEEE Computer Society 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.

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