American Impact Review

Original Research

A model for integrating regulatory requirements into continuous software delivery in financial organizations

Irina Titova*

Head of PMO IT, Independent researcher

ORCID: 0009-0004-1731-4572

* Corresponding author: Irina Titova, ititova808@gmail.com

OPEN ACCESS

Citation:
Irina Titova (2026). A model for integrating regulatory requirements into continuous software delivery in financial organizations. *Am. Impact Rev.*
10.66308/air.e2026079

Received: August 14, 2026
Accepted: September 22, 2026
Published: September 22, 2026

DOI:
10.66308/air.e2026079

ISSN: 3071-124X

Abstract

Continuous software delivery requires financial organizations to reconcile frequent changes with regulatory controls. This study develops a model for systematically integrating regulatory requirements into delivery processes to support demonstrable compliance while accounting for delivery-time constraints. The methods combine comparative analysis of published US and European Union regulatory and supervisory documents, analysis of published research, and structural-functional modeling. Sources include the Digital Operational Resilience Act, Commission Delegated Regulation 2024/1774, Federal Financial Institutions Examination Council guidance, and New York State Department of Financial Services cybersecurity requirements. The proposed classification distinguishes the stage at which an action must be completed, the control object, and the status of its source. The methodological contribution connects this classification with rules for translating and measuring controls: applicability, fulfillment, and evidence sufficiency are assessed separately, while the measurement unit follows the control object and frequency. A release-approval comparison identifies evidence that can be shared across sources and assessments that must remain separate. Analysis of software packages, test data, and release approval establishes sequencing constraints and conditions for evidence reuse. Normal and emergency routes are differentiated through prerequisites and follow-up actions. The proposed metrics cover delivery speed and stability, control fulfillment, evidence completeness, and time spent on controls. The result is a theoretical and methodological model supported by analytical reasoning; implementation effects were not measured.

Keywords: continuous delivery, financial organizations, regulatory requirements, digital operational resilience, change management, DevSecOps, compliance-as-code, traceability

Introduction

When releasing a new software version, a financial organization must align engineering decisions with applicable security and change-management requirements. In continuous delivery, this means determining which checks must be completed before release, which decisions require responsible personnel, and which records must be retained for subsequent review. Continuous integration and continuous delivery are referred to as CI/CD. Continuous delivery means maintaining release readiness through a repeatable process; it does not mean automatically deploying every change without approval.

In the European Union (EU), the regulatory basis for this analysis is Regulation 2022/2554 on digital operational resilience for the financial sector, known as the Digital Operational Resilience Act (DORA),

supplemented by technical standards for information and communication technology (ICT) risk management [1, 2]. The US sources are the Federal Financial Institutions Examination Council (FFIEC) handbook Development, Acquisition, and Maintenance and selected application-security requirements of the New York State Department of Financial Services (NYDFS), 23 NYCRR Part 500 [3-5]. These documents differ in legal status and scope. In particular, FFIEC guidance describes practices considered in supervision; publication of the handbook does not itself introduce new requirements [4]. The comparison must therefore distinguish binding requirements, supervisory expectations, and proposed technical implementations.

Published research already offers approaches to integrating continuous development and compliance controls. Kellogg et al. implemented continuous verification of selected security-related source-code properties and incorporated the results into auditing [6]. Moyón et al. proposed a model for integrating International Electrotechnical Commission (IEC) standard 62443-4-1 with a scaled agile development process [7]. Angermeir et al. organized the challenges of continuous security compliance and proposed a research agenda for automation [8]. Nweke and Yeng and Zakharchenko address requirements traceability, evidence management, and evidence generation during delivery [9, 10]. The systematic review by Rajapakse et al. shows that adopting development, security, and operations (DevSecOps) involves practices, infrastructure, and organizational factors as well as tools [11].

Gallina et al. use a multilayer ontology to describe relationships among legislation, standards, organizational processes, and compliance justification [12]. In the financial sector, Kurii et al. examine how DORA provisions relate to stages of the secure software development lifecycle [13]. Requirements integration, traceability, and the allocation of security measures to lifecycle stages are therefore established topics in published research.

A more specific question remains for control-point design: how can a requirement-to-stage mapping be translated into a release condition for a particular software version while retaining the regulated entity, responsibility, completion point, and grounds for exceptions? This study addresses that question for selected US and EU sources. Their selection permits a comparison of binding European provisions with US binding provisions and supervisory expectations concerning similar development and deployment activities. The comparison concerns conditions attached to individual actions, not the overall strictness of the two regulatory systems. Applicability, control placement, evidence, and measurement are specified together to distinguish reuse of technical materials from separate assessment under each regulatory basis.

Research objective

The study develops a model for systematically integrating regulatory requirements into continuous software delivery in financial organizations, aimed at supporting demonstrable compliance while accounting for delivery-time constraints in control design.

The research tasks are to:

1. Classify regulatory requirements and supervisory expectations concerning software development, deployment, and operation by the stage at which the controlled action must be completed, the status of its basis, and the control object.

2. Develop a model for translating the classified requirements into control points in a continuous delivery pipeline.

3. Propose metrics for assessing compliance and delivery speed.

Research contribution

The methodological contribution is the coordination of three control-design rules: assign a requirement to the stage at which its specific action must be completed; assess applicability, fulfillment, and evidence sufficiency separately; and select the measurement unit according to the control object and frequency. These rules are applied to US and EU provisions and supplemented by an analysis of temporal dependencies. The result identifies which release records can be shared, which decisions remain separate, and which conditions must be preserved when checks are rescheduled.

Materials and methods

This is a theoretical and methodological study based on published research and publicly available regulatory and supervisory documents. The analysis develops a control-design model and examines its internal consistency.

Regulatory materials were selected purposively. Included provisions directly concern the development, testing, approval, deployment, and maintenance of software changes. Sources with an explicit scope covering US or EU financial organizations were identified first. Provisions were then selected where a control object, required action, and completion point or frequency could be established. Scope and exemption provisions were included as conditions for interpreting these actions, not as standalone build checks. Table 1 presents the documents and their analytical roles. This is a bounded comparative analysis, not an exhaustive review of US and EU financial regulation.

Table 1. Sources and their roles in model development

Table 1. Sources and their roles in model development

Source	Status and scope	Material analyzed	Role in the model
DORA, Regulation (EU) 2022/2554 [1]	Binding for entities within its scope, subject to exclusions and the applicable regime	Article 9, paragraph 4, point (e): change management	Baseline requirement for a controlled change process
Commission Delegated Regulation (EU) 2024/1774 [2]	Supplements DORA; selected Title II provisions, outside the simplified Title III framework	Articles 10, 16, and 17: vulnerabilities, patches, development, testing, and changes	Control conditions, timing, responsibility, and exceptions
FFIEC Development, Acquisition, and Maintenance, 2024 [3, 4]	Supervisory guidance, not a standalone law	Development Roles; Testing; DevSecOps; Implementing Changes; Emergency Modifications; Change Management Documentation	Comparison of development, deployment, and follow-up controls
NYDFS 23 NYCRR Part 500 [5]	Binding on covered entities defined in Section 500.1, paragraph (e), subject to Section 500.19 exemptions	Section 500.7, paragraph (a), subparagraphs (1)-(4): access; Section 500.8: application security	Separation of access controls, development procedures, and periodic reviews
NIST SP 800-218, SSDF 1.1 [14]	Published engineering guidance; not treated here as independent financial regulation	Development organization, software protection, code analysis, and vulnerability response	Possible technical implementations of controls

Note: NIST denotes the National Institute of Standards and Technology; SSDF denotes the Secure Software

Development Framework. Source status is retained when selecting an implementation.

The main comparison concerns European provisions and US sources; NIST provides an engineering basis for implementation. International standards are not treated as a separate jurisdiction. The corpus excludes Sarbanes-Oxley Act financial-reporting requirements, specialized personal-data regimes, artificial-intelligence regulation, and comprehensive service-provider regulation.

The conditions of binding requirements and the content of supervisory expectations were established from the relevant regulatory and supervisory texts. Research publications were used to compare scholarly approaches.

The unit of analysis is an individual provision or a distinct supervisory expectation. Each record identifies the source and version, regulated entity, control object, required action, completion point, and conditions or exceptions. A complex provision is separated into actions where responsible parties, control points, or evidence differ. For example, prerequisites for an emergency change and its subsequent approval are analyzed separately even when grouped in one comparison row. The proposed implementation is recorded separately from the source's content. Table 2 groups related actions and is not used to count requirements.

Analysis and modeling followed five steps:

1. Identify provisions concerning the software-change lifecycle and establish their status.
2. Determine the primary completion stage, secondary links, source status, control object, and timing conditions.
3. Propose control procedures, verifiable conditions, and supporting evidence.
4. Compare shared procedures under different bases and identify temporal dependencies and measurement data.
5. Check the mappings analytically against three criteria: whether the control preserves the required action and deadline; whether its evidence supports an assessment for the correct object; and whether the measurement unit matches that object and the control frequency.

The consistency check compared cases with different conditions: independent approval, a test-data exception, periodic procedure review, and an emergency change. Where a successful technical operation did not answer the question posed by the source provision, a separate condition and supporting evidence were added. Where a requirement concerned the organizational process, it was not converted into an obligation to repeat that procedure for every release. This analysis examines internal consistency across differing application conditions.

Research publications were selected for their direct relevance to compliance automation, integration of requirements into development, and evidence management [6-13]. They were compared by analytical object, the relationship established between a requirement and a process, and the purpose of evidence. Only published studies were included; author copies of published articles were also used to access their content. This purposive selection supports comparison with directly related approaches.

In this model, demonstrable compliance means the ability to substantiate fulfillment of specific applicable requirements through verifiable evidence. Successful pipeline execution is not equated with a legal determination of organization-wide compliance. Unless qualified, DORA denotes the EU regulation; the DevOps Research and Assessment program is named in full when discussing delivery metrics.

Results

1. Classification by control point

The classification uses three independent attributes: the stage at which an action must be completed, the status of its basis, and the control object. Deadlines and frequency are specified separately. This distinction separates requirements that apply at the same stage but have different bases and measurement units.

Five stage classes are defined: **C1**, planning and maintenance of process rules; **C2**, development, build, and integration; **C3**, testing and acceptance of the integrated build; **C4**, approval and deployment; and **C5**, operation and feedback. This is a lifecycle design scheme whose content lies in the assignment rules, rather than in the phase names themselves.

The primary stage is where the controlled action must be completed. Input preparation and subsequent use of its results form secondary links. If a provision specifies actions with different completion points, each receives its own primary stage. Emergency prerequisites therefore belong to C4, while subsequent evaluation and approval belong to C5. Package security testing, R3, belongs to C2 because it must be completed no later than integration. Its secondary C3 link represents use of the results during later testing and acceptance; it does not move the check's deadline to that stage.

Table 2 uses **B** for a binding requirement and **S** for a supervisory expectation. An engineering practice, such as a NIST recommendation, informs implementation without becoming a binding requirement merely because it is included in a pipeline. The object is coded **Change** for a particular change, its code, components, or build, and **Proc.** for an organizational procedure, its application, or its periodic review. The object code describes the proposed control record: an obligation to maintain a procedure can generate both currency checks and records of its application to particular changes.

For example, the profile "C4; B; Change" concerns a particular change at approval or deployment under a binding provision. "C1; B; Proc." concerns process rules, not repetition of one organizational duty at every release. Secondary stages appear in parentheses. Responsibility and access separation are specified within the control, while applicability is determined before execution.

The classification preserves the distinction between a shared procedure and an individual decision. Acceptance and approval may use the same test results but answer different questions: whether the change is ready and whether the relevant function is authorized to permit deployment. Similarly, a shared code-analysis tool does not make the legal bases for its use identical. These distinctions determine the evidence and metric denominators used below.

2. Translating requirements into control points

The model connects two levels: defining control rules and executing a particular change. At the first level, responsible specialists determine source applicability, interpret the requirement, and approve assessment criteria. At the second, the pipeline executes the prescribed checks, obtains the necessary decisions, and retains records for a particular software version.

Figure 1 presents the model. Applicability and requirement content are established before a control procedure is selected; execution results are linked to evidence and metrics. Feedback informs rule revision through the established approval process.

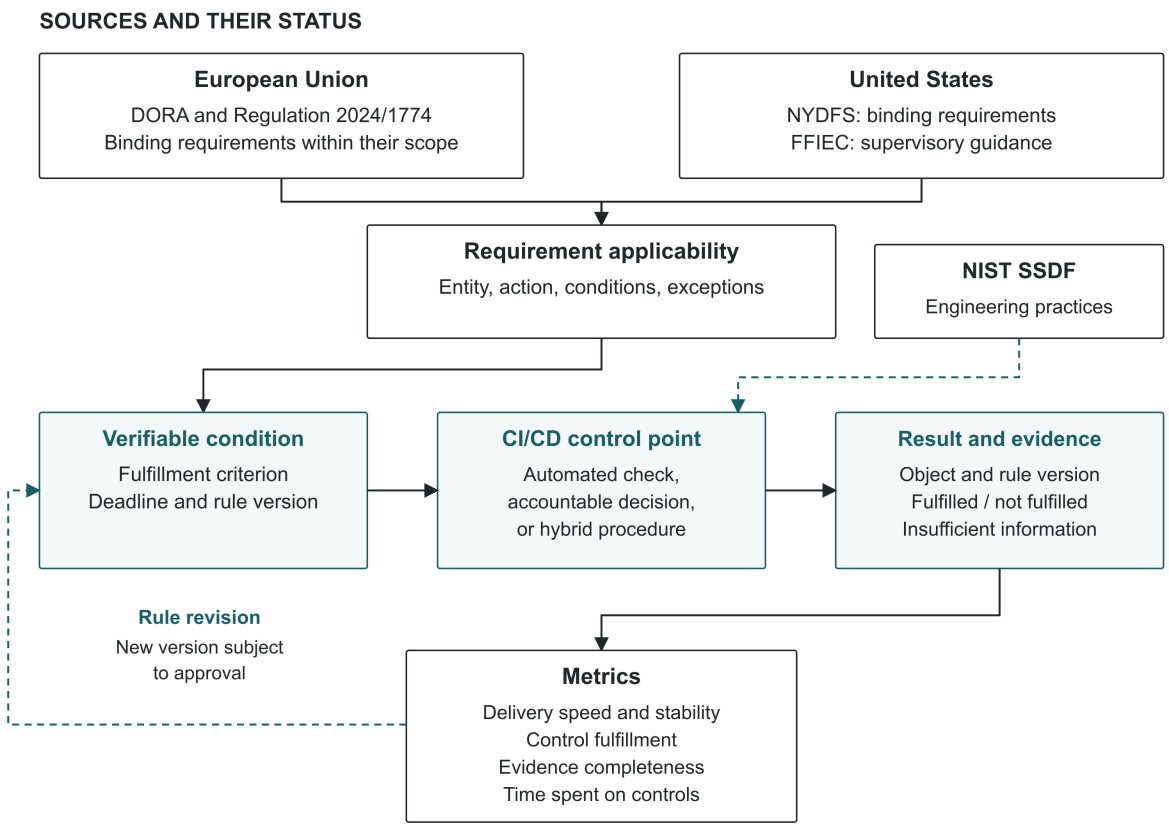

Figure 1. Integrating regulatory requirements into continuous software delivery.

Note: Solid arrows show the main sequence. Dashed arrows indicate engineering input and feedback for approved rule revision. Developed by the author from sources [1-5, 14].

Each control record is proposed to contain a requirement identifier; source provision; applicability basis; control object; primary stage and secondary links; deadline or frequency; performer and decision-maker; success criterion; verification method; evidence identifiers; and arrangements for negative results and permitted exceptions. Change controls identify the build version; procedure controls identify the procedure version and review period. Control-rule changes also require versioning and approval. Otherwise, an identical check status at different times may represent different conditions.

The model follows four principles.

Preserve the requirement's meaning

A technical check must address the question in the source provision. Checking that a report exists establishes its presence, not the acceptability of the defects it reports. Likewise, different user accounts alone do not establish appropriate functional separation.

Select the control method on a reasoned basis

Automation is used where a verifiable rule and reliable inputs can be specified. Decisions requiring substantive judgment are assigned to authorized personnel. A hybrid procedure combines machine verification with an accountable decision. Neither mandatory manual approval of every change nor universal replacement of

approval by automation is assumed.

Link evidence to the assessed object

Reports, approvals, and deployment records must identify the relevant change, build, and control-rule versions; periodic-review records must identify the relevant procedure and period. The model distinguishes fulfillment, failure, and insufficient information. Missing or outdated results do not count as success. Unresolved applicability is not recorded as non-applicability; it requires a separate decision by the responsible specialist.

Account for temporal dependencies

Each control separates a source-imposed constraint from the decision about its position in the process. The former specifies the required completion point; the latter determines the start, inputs, and potential for parallel execution. Table 3 develops these scheduling conditions.

Table 2 applies the classification to selected provisions. Profiles begin with the primary stage; grouped actions receive stage-specific labels. Row identifiers are reused in Table 3. Control and evidence descriptions are proposed implementations, not verbatim regulatory requirements. Commission Delegated Regulation 2024/1774 is abbreviated as CDR below.

Table 2. Classification and translation of selected provisions

Table 2. Classification and translation of selected provisions

ID and source condition	Profile	Proposed control	Evidence
R1. DORA, Article 9, paragraph 4, point (e): change management [1]	C1; B; Proc. (C2-C5)	Approve rules and link records across change stages	Procedure version; change record
R2. CDR, Article 16, paragraph 7: source-code integrity [2]	C2; B; Change	Check the origin and integrity of the code version	Version history; integrity-check results
R3. CDR, Article 16, paragraph 4: package security testing no later than integration [2]	C2; B; Change (C3)	Obtain results before the relevant integration is completed	Package version; report; check timestamp
R4. CDR, Article 16, paragraphs 3 and 8: pre-production code analysis and testing, with the specified applicability [2]	C3; B; Change (C2)	Link applicable analyses, findings, and decisions to the software version	Reports; issue decisions; analysis coverage
R5. CDR, Article 16, paragraphs 5-6: test data and exception conditions [2]	C3; B; Change	Check data preparation or conditions of the specific exception	Data description; exception period, approval, and notification
R6. CDR, Article 17, paragraph 1, point (b): independence of the approving function [2]	C4; B; Change	Compare requesting, implementing, and approving functions	Function and authority matrix; version-specific decision
R7. CDR, Article 17, paragraph 1, point (f): emergency safeguards [2]	C4; B; Change (C1)	Apply the approved emergency procedure and record prerequisites	Route justification; responsible parties; safeguards performed
R8. CDR, Article 17, paragraph 1, point (g): post-implementation emergency actions [2]	C5; B; Change	Track documentation, re-evaluation, assessment, and approval	Separate follow-up records

Continued on next page

ID and source condition	Profile	Proposed control	Evidence
R9. CDR, Article 10, paragraph 2 and paragraph 4, point (d): vulnerabilities and patch deadlines [2]	C5; B; Proc.	Track vulnerabilities, remediation, and established patch deadlines	Asset and vulnerability registers; remediation results; escalations
R10. FFIEC, DevSecOps (V.C.2), pp. 100-104: code and build controls [3]	C2; S; Change (C3, C4)	Link code, build, activity logs, and prescribed signatures	Artifact identifiers; logs; signature-verification results
R11. FFIEC, Testing (V.B), pp. 96-99 [3]	C3; S; Change	Link tests to the prepared version	Test plan and results
R12. FFIEC, Implementing Changes (VII.B.1), pp. 129-132: approval and post-implementation verification [3]	C4: approval; C5: verification; S; Change	Record the decision and compare the deployed change with the authorized change	Decision; deployment record; final verification
R13. FFIEC, Emergency Modifications (VII.B.3(c)), pp. 137-139 [3]	C4: prerequisites; C5: follow-up; S; Change	Check authority, abbreviated-testing risk, and subsequent route review	Decision; recovery measures; urgency assessment; documentation
R14. NYDFS, Section 500.7, paragraph (a), subparagraphs (1)-(4): access restrictions and review [5]	C1; B; Proc. (C2, C4)	Maintain necessary privileges and match them to change participants	Access rules; assigned privileges; review record
R15. NYDFS, Section 500.8, paragraph (a): security of internal and external applications [5]	C1; B; Proc. (C2, C3)	Select development and assessment procedures by application type	Approved procedures; application records
R16. NYDFS, Section 500.8, paragraph (b): application-security procedure review [5]	C1; B; Proc.	Track the prescribed review independently of releases	Procedure version; review decision and date

Note: B = binding requirement; S = supervisory expectation; Change = a particular change or software artifact; Proc. = organizational procedure. C1-C5 are defined in Section 1; parentheses indicate secondary stages. Grouped rows are not counts of requirements.

The mapping identifies two kinds of links. Some evidence follows a particular change from development into operation: a version identifier, for example, connects integrity checks, testing, and approval. Other evidence is maintained at process level and used by many changes, such as an authority matrix, secure development procedure, or vulnerability-remediation procedure. Reusing those materials does not create another instance of their review.

Shared procedure, separate bases

The comparison uses a common release-approval procedure. The model defines a **release evidence package** containing change and build identifiers, affected systems, requesters and implementers, approving functions and authority, a timestamped decision, check results, and the deployment record. Each applicable source is then assessed to determine which claims this package can substantiate.

Under point (b) of paragraph 1 of Article 17 of Commission Delegated Regulation (EU) 2024/1774, the approving function must be assessed for independence from the requesting and implementing functions [2]. The package therefore needs functional assignments, not just user names. Two different accounts within the same implementing function do not sufficiently establish independence. Approval and technical readiness are assessed separately: decision-making authority does not replace test results.

For FFIEC, the assessment concerns the appropriate decision-making level and the prescribed change-

management process, including subsequent verification of implementation (Implementing Changes, VII.B.1, pp. 129-132). Separation of duties is also addressed in Development Roles (II.B.3, p. 10), including role allocation in development and implementation and compensating measures for smaller organizations [3]. The shared record is therefore supplemented by the basis of authority and the organizational controls applied. This approach cannot automatically be treated as equivalent to the European independence condition.

Assessment under the NYDFS access-restriction and review requirements uses information about participants' privileges. These requirements appear in subparagraphs (1)-(4) of paragraph (a) of Section 500.7 [5]. This is a related but distinct basis: appropriate access supports the release procedure without replacing functional-independence assessment. Additional evidence concerns the privileges actually granted, their business necessity, and their latest review.

The comparative result is **a shared procedure with separate assessments**. One package may contain evidence for several bases, but the outcome for each remains separate, together with applicability and missing information. The analysis does not assume that all three sources apply simultaneously to one organization. It identifies which fields remain useful across bases and which criteria must be added.

A second case concerns the test-data exception. Paragraph 6 of Article 16 of Commission Delegated Regulation (EU) 2024/1774 links the exception to specific testing, a limited period, approval by the relevant function, and notification of the ICT risk-management function [2]. Even a complete successful test report therefore cannot replace evidence that the data use was permitted. The model records the dataset, purpose, permitted period, decision, and notification separately. Expiry or a change in the approved purpose requires reconsideration rather than automatic transfer of permission to the next test.

Timing constraints and control sequence

Delivery time is addressed by distinguishing a **source-imposed constraint** from a **model scheduling decision**. The former identifies the event by which an action must be completed or its required frequency. The latter determines when inputs can be prepared, which activities can overlap, and which changes require a repeated check. Table 3 presents these attributes.

Table 3. Timing conditions and control scheduling

Continued on next page

Control	Source-imposed constraint	Model scheduling	Evidence reuse

Table 3. Timing conditions and control scheduling

Control	Source-imposed constraint	Model scheduling	Evidence reuse
Code integrity, R2	CDR Article 16, paragraph 7, requires integrity controls without specifying a single check point [2]	Check when a version changes or is transferred; verify identity on subsequent use	Unchanged code and a verified chain of transfer
Package security, R3	No later than integration: CDR Article 16, paragraph 4 [2]	Start once the package version is known; independent test-data preparation may run in parallel	Same package, analysis rules, and validity conditions
Test data, R5	Exception limited to specific testing and a limited period: CDR Article 16, paragraph 6 [2]	Prepare data and assess the exception before use; the test waits for these conditions	Only within the approved purpose, dataset, and period
Pre-production analysis, R4	Before deployment: own code and, where feasible, third-party and open-source code; CDR Article 16, paragraph 8 [2]	Generate reports when required inputs are ready; check coverage and currency before release	Unchanged assessed object and material analysis conditions
Approval, R6, R12	Independent function under CDR; prior decision for normal changes under FFIEC, p. 131 [2, 3]	Prepare the authority matrix in advance; tie the decision to the version and required results	Matrix valid until authority changes; no automatic transfer of approval to a new version
Emergency follow-up, R8, R13	CDR: after implementation, without a universal numeric deadline; FFIEC: assessment and documentation as soon as possible, pp. 138-139 [2, 3]	Define responsibility and internal deadlines in the procedure; track follow-up separately	Separate records for each emergency change
Periodic review, R16	At least annually by the chief information security officer or a qualified designee: NYDFS Section 500.8, paragraph (b) [5]	Schedule independently of release count; use the current procedure at release	Review record belongs to a procedure and period, not to every release

Note: Reuse conditions are proposed design constraints, not general regulatory permission to omit checks.

The final column specifies model constraints, not a general authorization from regulators to skip testing. Matching versions is necessary but may be insufficient: changes to rules, environments, dependencies, or vulnerability information can invalidate an earlier result. For each report type, an organization specifies verifiable validity conditions and events that trigger reassessment.

The table supports a sequence for a normal change involving an external library and prepared test data. Once the library version is identified, security testing can begin. Dataset preparation and checks of participants' authority can proceed independently. Package testing must finish no later than integration; tests using the prepared data begin only after its use conditions are satisfied. The release decision concerns an identified version and the required results, followed by deployment and post-implementation verification.

This sequence permits independent preparation to overlap while preserving mandatory dependencies. Software-package security testing must be completed no later than integration, as specified in paragraph 4 of Article 16 of Commission Delegated Regulation (EU) 2024/1774 [2]. Approval cannot simply be assumed valid for a modified build without checking whether the decision covers it. The analytical result is an admissible order of actions: preparation and coordination can be reorganized while preserving the conditions attached to progression to the next stage.

Normal and emergency routes

In the normal route, preparation and testing are followed by verification of release conditions and the necessary approval. C5 remains after deployment: the change outcome is verified, monitoring continues, and identified problems are addressed. The emergency route additionally specifies the justification for urgency, authority, prerequisites, and follow-up actions.

Point (f) of paragraph 1 of Article 17 of Commission Delegated Regulation (EU) 2024/1774 requires procedures, tools, and protocols providing adequate safeguards for emergency changes. Particular safeguards are determined with reference to applicable provisions, including security-requirement verification and fallback procedures under points (a) and (e) of the same paragraph. Point (g) separately requires documentation, re-evaluation, assessment, and approval after implementation, including for workarounds and patches [2]. These actions cannot be reduced to closing a technical ticket.

FFIEC identifies personnel authorized to approve emergency changes and discusses risk-based decisions about testing scope. Where testing is abbreviated, backups and a back-out plan before implementation receive particular emphasis. Afterwards, the emergency classification and documentation are reviewed; assessment and documentation are completed as soon as possible (pp. 131, 138-139) [3].

In the model, the organization assigns responsibility for the preliminary decision, implementation, and follow-up. The European requirement for independence of the approving function is retained; FFIEC's authority arrangements are assessed separately. A specific internal follow-up deadline is a procedure parameter. Deferring an action until after release requires a basis in the applicable source; an exception in one regime does not permit bypassing a condition in another.

Figure 2 shows a shared process with a branch. Differences appear in prerequisites and C5 activities. The pre-deployment decision concerns conditions that must be satisfied by that point: a future obligation is neither marked overdue prematurely nor removed from tracking after release.

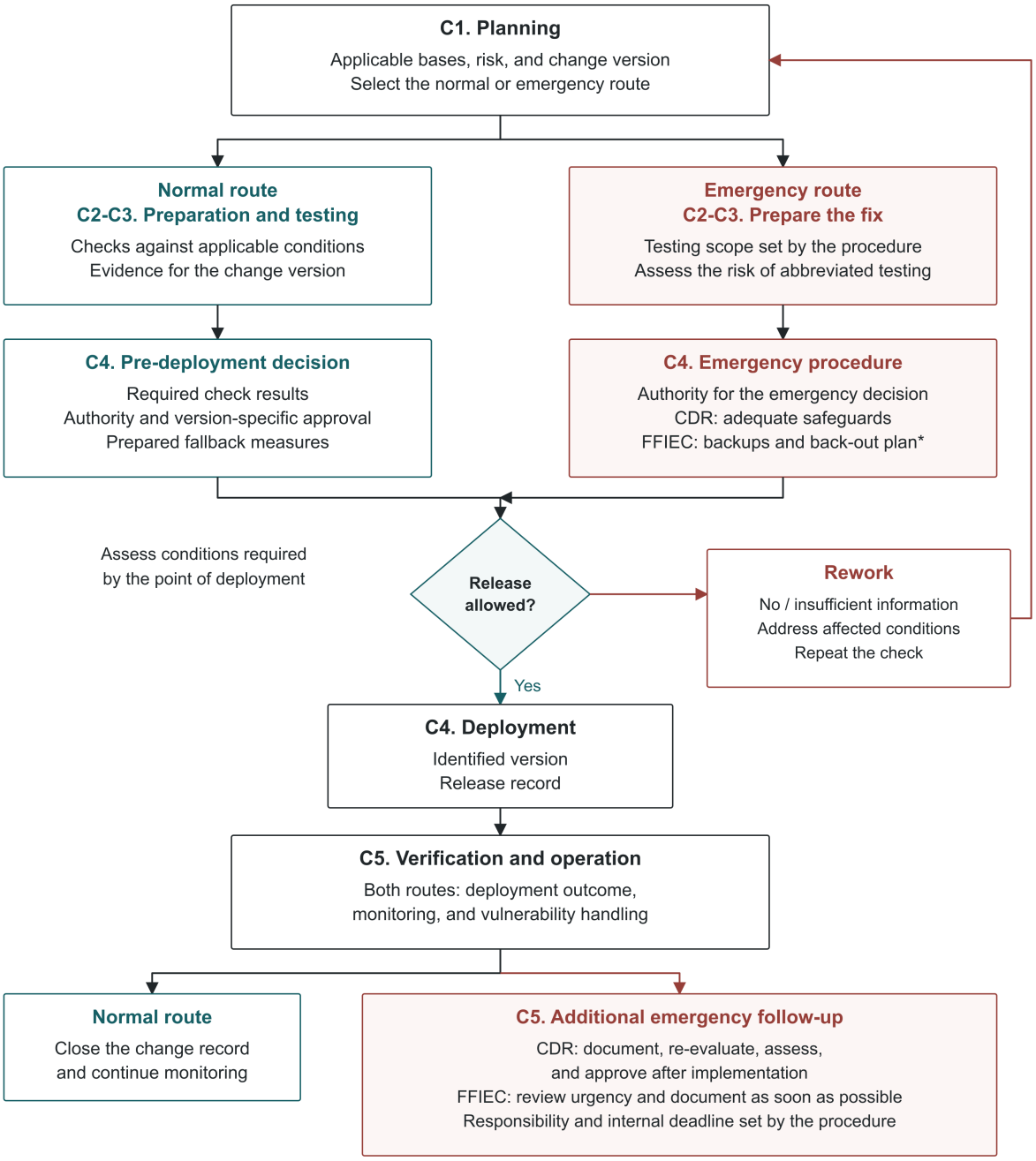

Figure 2. Shared process and differences between normal and emergency change routes.

Note: The diagram represents dependencies, not activity durations. Source-specific conditions are applied separately; a shared block does not imply legal equivalence. Emergency follow-up supplements normal operational verification. Developed by the author from sources [1-3, 5, 14].

3. Metrics for compliance and delivery performance

The proposed system combines three groups: delivery speed and stability; control fulfillment and evidence quality; and time-related and route-specific measures. Table 4 specifies the data and calculation rules for applying the model.

The first group follows the definitions of the DevOps Research and Assessment program [15]. The second and third provide operational measures of the proposed model. Their purpose is not to produce a single "percentage of legality," but to identify specific outcomes: an unchecked condition, insufficient evidence, a pending decision, or an incomplete emergency procedure.

Before calculation, the observation period, application or service, change and deployment boundaries, assessment time, rule version, and applicable control set must be defined. Binding requirements, supervisory expectations, and internal practices are measured separately.

For control fulfillment, the unit is a scheduled instance of a particular control for a specified object by a defined deadline. For release controls, this is a change-version/control pair; for periodic controls, it is a procedure/review-period pair. These sets are assessed separately. Rerunning a check does not create another successful instance, and an annual review is not multiplied by the number of releases that year. If one technical report supports several regulatory bases, the report may be reused, but fulfillment is assessed against each basis's own conditions. The number of bases is not interpreted as a count of independent checks.

Table 4. Metrics for assessing the model

Table 4. Metrics for assessing the model

Group and metric	Measurement rule	Interpretation and limitation
Speed: change lead time	From code commit in version control to production deployment; median and 90th percentile over the period	Process duration, not the cause of delay [15]
Speed: deployment frequency	Production deployments per period for one service	Compare using consistent deployment-counting rules [15]
Stability: change fail rate	Share of deployments requiring immediate intervention, remediation, or rollback	Delivery instability, not all operational incidents [15]
Stability: failed deployment recovery time	Time to recover from a deployment requiring immediate intervention	Not recovery time for incidents of every origin [15]
Stability: deployment rework rate	Share of unplanned deployments resulting from a production incident	Not equivalent to all emergency changes [15]
Control fulfillment	Scheduled control instances with verified fulfillment by the deadline / all applicable instances whose deadlines have arrived	Separate release and periodic controls; successful checks cannot offset a critical unmet condition
Evidence completeness	Instances in the same set with complete, current evidence at assessment / all instances in that set	Separate from success: a complete report can document failure
Release-condition violations	Deployed changes with at least one unmet binding pre-deployment condition	Record insufficient-evidence releases separately; do not count them as verified compliant
Decision waiting time	From readiness of the prescribed evidence package to the recorded decision; median and 90th percentile	Decision waiting, not total control duration
Share of lead time spent on controls	Duration of the union of mandatory-control execution and waiting intervals within lead time / lead time	Count overlaps once; not an estimate of delay caused by controls

Continued on next page

Group and metric	Measurement rule	Interpretation and limitation
Emergency-change share	Emergency-route changes / all changes in the period	Investigate increases; an increase alone does not establish circumvention
Follow-up completion	Emergency changes with all follow-up completed on time / emergency changes whose follow-up deadlines have arrived	Exclude recently deployed changes with deadlines still in the future

Note: Define scope and denominators before measurement. A zero denominator is reported as not applicable, not as complete fulfillment. The first five metrics follow [15]; the remaining definitions operationalize the proposed model.

Fulfillment and completeness measures require a predefined set of applicable controls. Removing failed checks from the denominator must not improve the result. Unresolved applicability is tracked separately; a metric for the resolved set does not describe those unresolved issues. An exception is admissible only where applicable rules provide a basis and all its conditions are evidenced. A management decision to accept risk does not waive a binding requirement. A zero denominator yields a not-applicable result rather than 100% fulfillment. Later remediation does not alter the historical assessment of deadline compliance.

For example, assessment of application-security procedure review under paragraph (b) of Section 500.8 of the NYDFS requirements examines timeliness, the responsible person's authority, and supporting records [5]. Successful releases under those procedures do not create additional completed reviews. Independent-approval controls, by contrast, concern particular changes. Mixing these units could allow numerous releases to conceal one overdue periodic control; a combined average across the two sets is therefore not proposed.

Assessment should distinguish normal and emergency routes, application types, and risk levels. Changes stopped by a control before release are tracked separately from deployed changes: successful prevention of a violation must not be conflated with a violation in an already completed deployment.

Parallel check durations cannot simply be added and treated as delivery delay. Two fully simultaneous ten-minute checks occupy ten minutes of elapsed time, although their summed durations equal twenty minutes. Even a correctly calculated union of intervals measures time occupied by controls, not the causal effect of introducing them. Evaluating an actual speed change in a subsequent implementation would require a comparable baseline process or an explicitly specified analytical dependency model.

No universal speed targets are prescribed. Required fulfillment conditions are not derived from average values of other metrics. Improving control organization and satisfying mandatory conditions are related but non-interchangeable tasks.

Discussion

Relationship to published approaches

The model develops established approaches to integrating requirements into development. Kellogg et al. examine code-property verification and its connection to auditing [6], while Moyón et al. connect a specific standard to a development process [7]. Here, the unit of analysis is an individual provision together with applicability, action, responsibility, and timing. Control placement, evidence, and measurement are jointly defined for that unit.

Traceability and machine-readable evidence are addressed by Nweke and Yeng [9], and continuous evidence collection is considered by Zakharchenko [10]. The contribution therefore lies in rules for specifying selected

provisions, not in inventing a requirement-control-evidence chain. Tables 2 and 3 show how a provision's content changes the control: integrity verification follows a code version, package testing is bounded by integration, and annual review concerns a procedure and period.

Gallina et al.'s ontology links legislation, standards, and organizational processes [12]. This study examines such links within the narrower problem of software change management in financial organizations. Kurii et al. map DORA to a secure development lifecycle [13]; the present analysis details selected provisions in terms of responsibility, exceptions, and frequency, and compares them with US sources. Regulatory conditions are established from primary documents.

The comparative result separates a shared procedure from assessments under different bases. Version, participant, and decision records can be shared, but the assessed condition changes: functional independence, appropriate management procedures, and necessary access are distinct. Reducing duplicate evidence collection therefore does not reduce the number of applicable obligations.

Relationship to COBIT 2019 and ITIL 4

Established change-management approaches already include organizational and informational components. An ISACA publication on COBIT 2019 uses the BAI06 Managed IT Changes objective to explain the structure of management objectives, including processes, organizational structures, information items, and metrics [16]. The official ITIL 4 Change Enablement description includes risk assessment, authorization, scheduling, roles, and metrics [17]. Responsibility, records, and metrics alone therefore do not distinguish this model from those approaches.

For release approval, the general management framework establishes the need for decisions and allocated responsibility. The model specifies the assessment for a selected basis: CDR requires examination of relationships among the approving, requesting, and implementing functions; the FFIEC assessment concerns authority and change-management arrangements. The resulting record identifies the condition assessed for a particular version and the evidence supporting the assessment.

For test data, general risk assessment is supplemented by the specific exception conditions in paragraph 6 of Article 16 of Commission Delegated Regulation (EU) 2024/1774: the use case, limited period, approval, and notification [2]. This detail demonstrates application of the model to a regulatory provision. It does not establish that equivalent mechanisms are absent from COBIT or ITIL.

The comparison is limited to publicly available official descriptions; the full BAI06 practices and detailed ITIL 4 practice guides were not analyzed. The distinction is therefore demonstrated at the level of the selected analytical unit and its application, without claiming superiority over these frameworks. The ITIL comparison explicitly concerns ITIL 4 rather than the entire publication family.

Practical implications and limitations

The model provides a control-design sequence: identify an applicable provision, isolate its action and completion point, select a verifiable condition, link the result to its object, and define a measurement unit. Its timing component separates waiting for necessary inputs from work that can be prepared in advance or performed concurrently. Pipeline and evidence-storage tools are selected during implementation.

Transfer to other regimes requires a new selection of provisions and applicability assessment. The European analysis covers selected CDR Title II provisions; the US analysis is not a complete inventory of federal and state obligations. In particular, application-security requirements under NYDFS Section 500.8 must be considered alongside exemptions. Section 500.19, paragraph (a), provides an exemption from Section

500.8 for entities meeting its conditions [5]. Differences in covered entities remain important even where engineering procedures are similar.

The limitations arise from purposive source selection and the analytical study design. Independent corpus coding, expert assessment, and measurement of implementation effects were not undertaken. The consistency analysis examines the proposed mappings but does not establish the legal sufficiency of a particular implementation, its acceptance by an auditor, or an actual reduction in delivery time. Further evaluation could involve specialist assessment of mapping completeness and measurement using process data.

Conclusions

1. Selected US and EU provisions were classified by action-completion stage, source status, and control object. The five stages are complemented by attributes that distinguish particular changes from organizational procedures. The translation table covers development, testing, approval, deployment, and operational controls.

2. A model was developed linking applicable provisions to verifiable conditions, responsibility, and evidence. The release-approval comparison shows how one evidence package can support separate CDR, FFIEC, and NYDFS assessments. Timing constraints and evidence-reuse conditions were specified for packages, test data, approval, and follow-up. Delivery time is thereby addressed through a reasoned control sequence, including differences between normal and emergency routes.

3. Metrics were proposed for delivery speed and stability, control fulfillment, evidence completeness, and time-related costs. Measurement units follow control object and frequency; the rules distinguish bases and address insufficient information, parallel checks, and deadlines that have not yet arrived.

The result is a theoretical and methodological model with analytical support from selected sources. Its actual effects on compliance and delivery speed require evaluation during application.

Declarations

Acknowledgments. Use of artificial intelligence. Codex (OpenAI) was used to verify the cited sources and reference details, to lay out and format the figures, and to translate the manuscript into English.

Author contributions. Irina Titova: Conceptualization; Methodology; Investigation; Formal analysis; Writing - original draft; Writing - review and editing.

Funding. This research received no external funding.

Conflicts of interest. The author declares no conflicts of interest.

Data availability. The study is based on publicly available published sources listed in the References.

Ethics approval. Not applicable. The study analyzes published documents and does not involve human participants, animals, or identifiable personal research data.

References

1. European Parliament and Council. Regulation (EU) 2022/2554 of 14 December 2022 on digital operational resilience for the financial sector. Official Journal of the European Union. 2022;L333:1-79. Available from: https://eur-lex.europa.eu/eli/reg/2022/2554/oj. Accessed July 23, 2026.

2. European Commission. Commission Delegated Regulation (EU) 2024/1774 of 13 March 2024 supplementing Regulation (EU) 2022/2554 with regard to regulatory technical standards specifying ICT risk management tools, methods, processes, and policies and the simplified ICT risk management framework. Official Journal of the European Union. 2024. Available from: https://eur-lex.europa.eu/eli/reg_del/2024/1774/oj. Accessed July 23, 2026.

3. Federal Financial Institutions Examination Council. Development, Acquisition, and Maintenance. Information Technology Examination Handbook. August 2024. Available from: https://ithandbook.ffiec.gov/it-booklets/development-acquisition-and-maintenance/. Consulted August 2024 PDF edition, hosted by Wolters Kluwer/CCH: https://business.cch.com/BFLD/FFIEC-IT-DevelopmentAcquisitionMaintenance-Hankbook-08292024.pdf. Accessed September 11, 2026.

4. Federal Deposit Insurance Corporation. Updated FFIEC IT Examination Handbook: Development, Acquisition, and Maintenance Booklet. FIL-60-2024. August 29, 2024. Available from: https://www.fdic.gov/news/financial-institution-letters/2024/updated-ffiec-it-examination-handbook-development. Accessed September 11, 2026.

5. New York State Department of Financial Services. 23 NYCRR Part 500: Cybersecurity Requirements for Financial Services Companies. Second Amendment, November 1, 2023. Informational text, not the official edition of the rule. Available from: https://www.dfs.ny.gov/cybersecurity/23-NYCRR-Part-500. Accessed September 11, 2026.

6. Kellogg M, Schäf M, Tasiran S, Ernst MD. Continuous compliance. In: Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering. 2020. p. 511-523. doi:10.1145/3324884.3416593.

7. Moyón F, Méndez Fernández D, Beckers K, Klepper S. How to integrate security compliance requirements with agile software engineering at scale? In: Product-Focused Software Process Improvement. PROFES 2020. 2020. p. 69-87. doi:10.1007/978-3-030-64148-1_5.

8. Angermeir F, Fischbach J, Moyón F, Mendez D. Towards automated continuous security compliance. In: Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. 2024. p. 440-446. doi:10.1145/3674805.3690748.

9. Nweke LO, Yeng PK. Compliance-as-code for AI-driven identity systems: clause-to-control traceability and machine-readable evidence. IEEE Access. 2026;14:28258-28281. doi:10.1109/ACCESS.2026.3665991.

10. Zakharchenko A. Integrating continuous compliance into DevSecOps pipelines: a data engineering perspective. Software. 2026;5(1):6. doi:10.3390/software5010006.

11. Rajapakse RN, Zahedi M, Babar MA, Shen H. Challenges and solutions when adopting DevSecOps: a systematic review. Information and Software Technology. 2022;141:106700. doi:10.1016/j.infsof.2021.106700.

12. Gallina B, Steierhoffer GL, Young Olesen T, Parajdi E, Aarup M. Towards an ontology for process compliance with the (machinery) legislations. Journal of Software: Evolution and Process. 2025;37(1):e2728. doi:10.1002/smr.2728.

13. Kurii Y, Susukailo V, Yerofeieva A, Nesteriuk M. Analysis of the impact of Digital Operational Resilience Act (DORA) requirements on the process of secure software development. Cybersecurity: Education, Science, Technique. 2025;4(28):400-412. Ukrainian. doi:10.28925/2663-4023.2025.28.852.

14. Souppaya M, Scarfone K, Dodson D. Secure Software Development Framework (SSDF) version 1.1: recommendations for mitigating the risk of software vulnerabilities. NIST SP 800-218. February 2022. doi:10.6028/NIST.SP.800-218.

15. Harvey N. DORA's software delivery performance metrics. DevOps Research and Assessment. Available from: https://dora.dev/guides/dora-metrics/. Accessed September 11, 2026.

16. Thomas M. A new COBIT is in town and I really like how it looks. ISACA Industry News. December 10, 2018. Available from: https://www.isaca.org/resources/news-and-trends/industry-news/2018/a-new-cobit-is-in-town-and-i-really-like-how-it-looks. Accessed September 11, 2026.

17. PeopleCert. ITIL 4 Practitioner: Change Enablement. Official practice and course overview. Available from: https://www.peoplecert.org/browse-certifications/it-governance-and-service-management/ITIL-1/itil-4-practitioner-change-enablement-3794. Accessed September 11, 2026.