NextGen Scientific Publication
American International Journal of Computer Science and Technology
Volume 6 Issue 4, Pg. No. 106-115, AIJCST-V6I4P110, 2024
https://doi.org/10.63282/3117-5481/AIJCST-V6I4P110

Original Article

Automated Regulatory Documentation Generation for IVDR and FDA Compliance

***Saket Mishra**

Director, Product Development, Illumina Inc., Foster City, California, USA.

Abstract:

Medical device software development is subject to strict regulatory regimes that require copious documentation of safety, efficacy, traceability and risk control. During the software development lifecycle many interrelated documents are generated that are required for regulatory submissions under the European in Vitro Diagnostic Regulation (IVDR), the United States Food and Drug Administration (FDA) and software lifecycle standards such as IEC 62304. We present a framework for automated generation of regulatory documentation that translates structured software engineering artefacts into draft regulatory documents, via traceability graph construction, template-based document synthesis and consistency validation. The proposed architecture combines requirement repositories, software development plat-forms, continuous integration pipelines, and software composition analysis outputs to automatically assemble documentation in compliance with IVDR Annex II, Annex III, FDA eSTAR submissions, and related regulatory structures. Analytical evaluation demonstrates improvements in documentation consistency, lifecycle traceability, artifact synchronization, and regulatory preparedness while preserving mandatory human review before final submission. The framework establishes a foundation for future intelligent regulatory automation capable of supporting evolving international compliance requirements.

Keywords:

Medical Device Software, IVDR, FDA, IEC 62304, Regulatory Documentation, Traceability, Software Life-cycle, Compliance Automation, Software Bill of Materials, Technical Documentation.

Article History:

Received: 04.06.2024

Revised: 02.07.2024

Accepted: 10.07.2024

Published: 22.07.2024

1. Introduction

The increased software complexity of medical devices to-day has significantly increased regulatory expectations on documentation of software development. Regulators require manufacturers to provide not only functional correctness but also full lifecycle evidence linking requirements, software architecture, implementation, verification, validation, cybersecurity and risk management. The introduction of the European in Vitro Diagnostic Regulation (IVDR), evolving FDA guidance on software and increased focus on cybersecurity documentation have all raised expectations.

Regulatory submissions consist of some interrelated documents such as software development plans, software architecture descriptions, verification reports, validation evidence, risk management files, cybersecurity documentation, software bills of materials (SBOMs), clinical evidence summaries, usability engineering reports, and post-market surveillance documentation. Although software engineering platforms increasingly maintain structured project information, the final regulatory dossiers are frequently assembled manually by quality assurance (QA) and regulatory affairs (RA) specialists.

Manual documentation generation presents several challenges. Project artifacts evolve continuously during iterative development, while regulatory documents are often updated periodically. Consequently, inconsistencies emerge between implemented functionality and documented evidence. Trace-ability links may become incomplete, verification records may no longer correspond to current requirements, and outdated software component inventories may remain within submission packages. Such inconsistencies increase internal review effort and complicate external regulatory assessment.

While commercial lifecycle management platforms provide extensive traceability management, they tend to focus on arte-fact storage and relationship visualisation, rather than automatically generating documentation that is ready for submission. Accordingly, a lot of manual effort is still necessary to translate engineering artefacts into structured regulatory dossiers based on authority-specific templates.

Directly synthesising regulatory documents from structured software artefacts, an automated documentation generation framework provides an opportunity to reduce the overhead of documentation maintenance while improving the consistency across evolving software projects. Such automation helps to prepare documentation by continuously keeping synchronised draft documentation based on validated engineering artefacts, rather than replacing the regulatory specialists.

The framework presented in this paper integrates requirements repositories, software testing outputs, continuous integration pipelines, software composition analysis reports, and risk management information into a unified traceability graph. Template-driven document synthesis generates structured draft documentation aligned with IVDR Annex II, Annex III, and FDA eSTAR requirements. Rule-based consistency validation further identifies incomplete traceability, outdated references, missing verification evidence, and lifecycle inconsistencies before regulatory review.

2. Regulatory Background

Medical device software is subject to many international standards and regulatory regimes that collectively require full technical documentation throughout the product life cycle. Different jurisdictions have different formats for submission. Typical documentation principles include traceability through-out the product lifecycle, evidence-based verification, risk management, cybersecurity assurance and design control. Annex II and Annex III of the European Union in Vitro Diagnostic Regulation (IVDR 2017/746) specify detailed technical documentation requirements. These annexes require documentation that describes device design, manufacturing in-formation, software validation, performance evaluation, risk management, post-market surveillance planning and life-cycle evidence. The documentation should be consistent for all development activities at all times.

The FDA software guidance and the electronic Submission Template and Resource (eSTAR) demand structured sub-mission packages, such as software description, architecture information, hazard analysis, verification evidence, cybersecurity documentation, software maintenance information, and software component inventories, developed according to the requirements of the FDA Quality System Regulation within the United States, 21 CFR Part 820. IEC 62304 is the international software lifecycle framework that governs software development processes, configuration management, software maintenance, verification, validation and risk control activities. The standard stresses the need for bidirectional traceability between system requirements, software requirements, architecture, implementation, testing, and maintenance records.

ISO 14971 is used in conjunction with IEC 62304 to define systematic risk management processes throughout the lifecycle of the medical device. Documentation must provide full traceability for all risk identification, hazard analysis, mitigation implementation, verification of control measures and residual risk evaluation. Recent cybersecurity guidance also requires documenting third-party software dependencies using Software Bills of Materials (SBOMs), vulnerability management procedures, secure development practices, and coordinated vulnerability disclosure mechanisms. Regulatory submissions are increasingly using standardised SBOM representations such as those provided by SPDX and CycloneDX frameworks. The documentation requirements share a significant struc-tural overlap, focused on lifecycle traceability and artefact consistency, though the terminology and submission format differ between them.

3. Related Work

Application Lifecycle Management (ALM) platforms have traditionally provided centralised repositories to support regulatory documentation management of engineering artefacts. Commercial platforms like IBM DOORS Next, Siemens Polarion ALM, Jama

Connect, Azure DevOps and Atlassian Jira offer rich capabilities for requirements management, issue tracking, test management and life-cycle traceability. These platforms enable users to define explicit relationships between software requirements, design specifications, implementation tasks, software tests, defects and change requests. Reporting modules also support the generation of project summaries and traceability matrices. While such capabilities exist, current platforms are mainly focused on artefact management, not on the automated generation of complete regulatory documentation. Regulatory specialists usually transfer project information to external templates and generate submission documents in the authority-specific structure manually.

Research works have studied document generation based on model-driven engineering, software traceability analysis, natural language processing, and requirements engineering automation. Automated traceability recovery techniques have used textual similarity analysis, information retrieval methods, graph-based linkage inference and semantic embeddings to find relationships between software artefacts. These approaches improve the lifecycle consistency but generally stop at the artefact linkage instead of completing the document generation. More recent work has explored knowledge graphs capturing software engineering artefacts and regulatory constraints. Graph representations enable impact analysis, visualisation of dependencies and consistency checking within the framework of evolving software projects. But, the direct mapping between graph representations and regulatory dossier structures is relatively unexplored.

Large Language Models (LLMs) have also become promising tools for technical document drafting. However, for regulatory submissions, deterministic traceability, reproducibility and evidence-based statements are required. Pure generative approaches thus raise concerns around unsupported content generation and lack of auditability [23]. The main research gap lies in the gap between lifecycle traceability management and the generation of submission-ready regulatory documentation. Current systems do a good job of managing engineering artefacts but do not provide much capability for continuously generating synchronised regulatory documents which reflect the current state of the project.

4. System Design and Architecture

The proposed framework converts structured software engineering artifacts into draft regulatory documentation through a modular processing pipeline comprising artifact extraction, normalization, traceability graph construction, template map-ping, document synthesis, and consistency validation.

Figure **??** illustrates the overall workflow.

The framework starts by collecting structured engineering artefacts from a variety of software development environments. Requirement Repositories like Jira, Azure DevOps or Polarion offer functional requirements, user stories, defects and change requests. Continuous integration pipelines deliver software verification results, unit testing reports, integration testing results and build metadata. Software composition analysis tools generate SBOMs according to the CycloneDX or SPDX specifications. Risk Management Repositories will provide hazard analyses and mitigation records in compliance with ISO 14971.

The normalisation process converts every engineering arte-fact, regardless of the originating platform, into a common intermediate representation consisting of standardised identifiers, version metadata, timestamps, lifecycle states, ownership information and semantic classifications. This representation allows further graph construction without being tied to repos-itory specific formats.

The normalised artefacts form a traceability graph of engineering relationships across the software lifecycle. Nodes represent requirements, software modules, verification activi-ties, risks, software components, cybersecurity evidence, and regulatory sections. Directed edges represent relationships such as satisfies, verifies, mitigates, depends-on, implements and references. Then graph traversal algorithms fill the document templates for specific regulatory structures. IVDR Annex II, Annex III, FDA eSTAR sections, software lifecycle reports, cybersecurity documentation, technical file structures, all have configurable template schemas. Each template defines required evidence, required sections, supporting artefacts, and traceability constraints.

Generated documents are kept in sync with evolving engineering repositories through incremental updates triggered by software changes, requirement changes, verification completion or SBOM changes. Continuous synchronisation prevents much drift between engineering activities and regulatory documentation. A consistency validation module performs rule-based verification before document generation. Validation rules identify orphan requirements, incomplete verification coverage, obsolete software

component references, unresolved change impacts, inconsistent version identifiers, duplicate traceability paths, and missing regulatory evidence. The resulting validation report accompanies generated documentation to facilitate regulatory review.

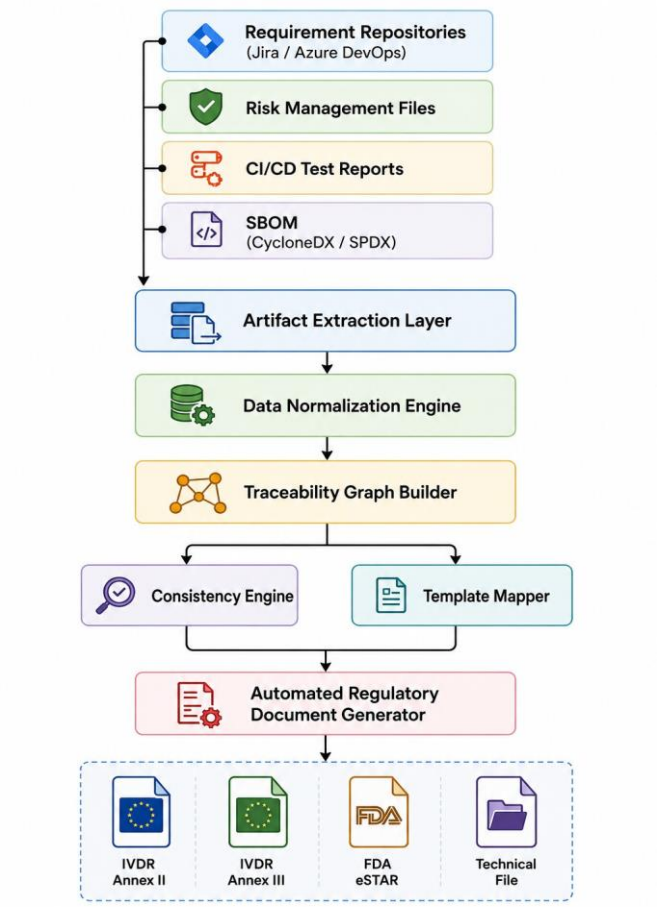

Figure 1. Overall Architecture of the Automated Regulatory Documentation Generation Framework

5. Traceability Graph Construction

The traceability graph is a core component of the proposed framework, which is an intermediate knowledge representation linking software engineering artefacts throughout the development lifecycle. The proposed graph captures multidimensional dependencies between regulatory evidence, software components, risk controls, verification activities, cybersecurity artefacts, and configuration information, unlike traditional traceability matrices that predominantly focus on pairwise relationships between requirements and tests. Each engineering artefact is represented as a node in the graph, with a set of standard meta-data: unique identifiers, artefact category, lifecycle status, version information, owner-ship, creation history, and repository source. Directed edges encode relationships between nodes and describe semantic associations relevant to regulatory compliance. Typical edge types are implements, verifies, mitigates, depends-on, references, supersedes and contributes-to.

After artefact normalisation the graph construction process is performed. In software repositories there are structured identifiers that can be used to link deterministically in case of explicit references. Relationships with high confidence are established between requirement identifiers found in software test cases, change requests, risk controls or design specifications without need of inference mechanisms. Semantic linkage inference is used to supplement graph construction for engineering artefacts lacking explicit references. NLP techniques to compare artefact descriptions based on their lexical similarity, domain terms, software component names, and contextual metadata. Candidate relationships, generated by semantic similarity, are verified against the repository metadata before being inserted into the graph. This hybrid strategy strikes a balance between deterministic traceability and automated relationship discovery, while minimising unsupported associations.

Version control information enhances graph connectivity by linking software revisions to the documentation updates. Every node keeps a history of relationships, which allows the re-building of documentation states for earlier releases of software. This temporal traceability supports regulatory audits that require to show the development history across different product versions. Change impact analysis operates on graph connectivity directly. When a change is made to requirements, software architecture, software components, and verification evidence, dependency traversal algorithms are used to identify down-stream artefacts that need to be updated in the documentation. Incremental updates keep documents consistent, and save unnecessary processing by not regenerating entire dossiers.

The graph representation additionally supports visualization of documentation completeness. Orphan nodes lacking in-coming or outgoing regulatory relationships indicate potential documentation deficiencies, whereas disconnected subgraphs may reveal incomplete lifecycle evidence requiring additional engineering activities before regulatory submission.

Table 1. Comparison of Existing Lifecycle Management Platforms and Automated Documentation Framework

Capability	Traditional ALM Plat-forms	Traceability Tools	Document Templates	Proposed Framework
Requirement Management	Supported	Supported	Limited	Supported
Traceability Visualization	Comprehensive	Comprehensive	Limited	Comprehensive
Automatic Document Gen-eration	Minimal	Minimal	Template-Based	Integrated
Cross-Repository Integration	Partial	Partial	Limited	Unified
Consistency Validation	Basic	Moderate	None	Rule-Based
SBOM Integration	Limited	Limited	None	Native
Continuous Synchronization	Limited	Limited	Manual	Supported
Framework-Specific Templates	External	External	Supported	Integrated

5. Consistency and Validation Checks

The regulatory readiness from automation of document generation will only be as good as the underlying engineering artefacts are consistent with the requirements of the applicable standards. To this end, the proposed framework includes a dedicated validation engine which performs rule-based checks before document synthesis. The validation rules are grouped by the phases of the software lifecycle, so that the engineering completeness of each phase can be systematically checked. Requirements engineering, software implementation, verification and validation, cybersecurity, risk management and configuration management.

Requirement validation ensures that all accepted software requirements have evidence of implementation, verification procedures, and risk links where applicable. Requirements without downstream evidence are flagged for further engineering review prior to inclusion in the regulatory documentation. Verification consistency ties the functional requirements to the software tests. We use graph traversal to find missing verification records, duplicate test mappings, obsolete validation reports and unresolved software defects. As the verification evidence is likely to evolve during the iterative development process, the continuous synchronisation helps to keep the documentation in line with the current status of testing.

Risk management validation provides the evidence that links the identified hazards, the mitigations that have been put in place, the verification evidence and the residual risk assessments. Validation results that need review are due to unverified mitigation

controls, old hazard analyses, or lack of complete linkage between software functionality and related risks. Cybersecurity validation includes Software Bill of Materials (SBOM) analysis. By comparing software dependencies extracted from CycloneDX or SPDX representations with the current software repositories, obsolete component references, missing dependency declarations, inconsistent version identifiers or undocumented third-party libraries are identified.

Validation of configuration management ensures that soft-ware versions match documentation revisions, repository branches, software builds, and release identifiers. Automated validation is an important part of manual documentation work-flows as mismatches between software releases and referenced documentation are not uncommon. The validation engine also supports cross-version comparison in iterative software development. Graph differencing algorithms can detect changes in the regulatory evidence between the consecutive releases of the software instead of evaluating the documentation of each release separately. Documentation updates continue to focus on changed engineering artefacts but include regulatory evidence from previous releases that have not changed. The validation results are aggregated by severity and by lifecycle domain. Instead of automatically updating engineer-ing artefacts, the framework creates structured review reports, enabling quality assurance and regulatory specialists to review discovered inconsistencies prior to regulatory submission.

6. Evaluation Methodology

Analytical assessment of typical workflows used in documentation during the development of software for medical devices is used to evaluate the proposed framework. The evaluation is on process characteristics, documentation quality attributes, traceability completeness, maintainability, and regulatory readiness, not on quantitative performance measures. The evaluation includes two typical documentation work-flows. The first is traditional documentation where engineering artefacts are kept separately and regulatory specialists manually assemble documentation for submissions from exported reports, spreadsheets and document templates. The second is the preparation of documentation based on the proposed automated framework combining engineering repositories and continuous documentation generation. Documentation consistency is the extent to which the regulatory documents generated are reflective of the current state of the engineering artefacts, without the need for repetitive manual synchronisation. Automating the synchronisation reduces the possibility that the software implementation diverges from the supporting documentation.

Traceability completeness measures lifecycle connectivity for requirements, software architecture, implementation, test-ing, risk management, cybersecurity documentation and regulatory evidence. End-to-end traceability facilitates internal quality assurance and external regulatory review. Maintainability is the effort required to keep documentation updated during iterative software development. Graph-based synchronisation with incremental updates simplifies the maintenance of documentation compared to repeated manual revision of independent documents. Regulatory readiness is an assessment of the generated documentation against organisational expectations for completeness, document structure, evidence of lifecycle activities and review readiness before formal submission. The rule-based validation also enhances readiness by identifying inconsistencies in documentation prior to being assessed by regulatory authorities.

Scalability refers to the ability of the framework to support heterogeneous engineering repositories and simultaneously support a number of regulatory templates. Software organisations often build products for several international markets. Documentation generation must accommodate multiple regulatory structures without duplicated engineering effort. Auditability is about preserving artefact provenance during document generation. All generated document sections are associated with source engineering artefacts, providing traceability for transparent review by regulatory agencies and facilitating verification of documentation accuracy. The proposed framework is not meant to replace existing quality assurance procedures but to act as an augmentation layer for the continuous synchronisation of draft documentation while the mandatory expert review prior to submission remains intact.

7. Results

The analysis shows that the automatic generation of documentation improves the consistency of the product lifecycle by continuously synchronising engineering repositories with regulatory documentation. The graph-based architecture creates traceability relationships that are persistent and resilient to iterative software evolution, which reduces the need for manual document reconciliation. The integrated extraction pipeline increases the complete-ness of the documentation by combining the engineering evidence from multiple software development environments into one regulatory knowledge representation. Template-driven synthesis informs the creation of coherent regulatory narratives by aggregating requirements repositories, software verification systems,

cybersecurity artefacts, and risk management documentation.

Continuous graph synchronisation improves documentation maintainability. Engineering changes are automatically propagated through traceability relationships, enabling the documentation to be updated to reflect the evolving software configurations. The consistency validation engine enhances documentation quality through early identification of lifecycle inconsistencies before the regulatory review. Rules for detecting orphan requirements, incomplete verification evidence, unresolved soft-ware dependencies, out-of-date configuration references and missing risk associations help to systematically refine documentation during development, not just before submission.

From a regulatory perspective, template-driven document synthesis supports the structural homogeneity of submission packages. Generation of regulatory sections from standardised engineering artefacts is more consistent in terms of terminology, artefact references, and organization of evidence across the lifecycle than documentation assembled manually from heterogeneous sources of information. The graph representation provides broader organisational benefits than just generating documents. Traceability visualisation supports the analysis of the impact of changes to requirements, software updates, disclosures of cybersecurity vulnerabilities, or re-evaluations of risks.

And the framework also enables regulatory harmonisation by separating engineering knowledge representation from jurisdiction-specific document templates. Therefore, joint engineering evidence can populate multiple regulatory structures such as IVDR Technical Documentation, FDA eSTAR submissions, IEC 62304 lifecycle reports and organization quality management documentation without duplicating engineering efforts. Automated document synthesis increases consistency and maintainability, but regulatory documentation still requires ex-pert interpretation. Summaries of clinical evidence, regulatory justification narratives, benefit-risk analyses and submission strategy still rely on domain expertise that extends beyond deterministic artifact transformation. In this sense, the pro-posed framework is orientated towards assisted documentation generation rather than autonomous regulatory submission preparation.

8. Discussion

The proposed framework demonstrates that regulatory documentation can be perceived as an ever-evolving software engineering artifact instead of a set of static documents pre-pared shortly before submission. By constructing an integrated traceability graph from structured engineering repositories, documentation generation is closely integrated with software development activities, thereby mitigating inconsistencies introduced through manual synchronization. Another important advantage of the proposed architecture is that it does not depend on the repository. Artifact normalization allows organizations to integrate heterogeneous development environments without requiring them to change their Application Lifecycle Management (ALM) platforms. Thus, organizations using Jira, Azure DevOps, Polarion, IBM DOORS Next gen, or custom requirement management systems can adopt this framework with minimum disruption to their existing engineering workflow.

However, there are a number of practical limitations to these advantages. The regulatory documents often have sections that require expert interpretation, not deterministic extraction from engineering repositories.

Table 2. Qualitative Comparison between Manual and Automated Regulatory Documentation Workflows

Evaluation Aspect	Conventional Manual Workflow	Proposed Automated Framework
Artifact Synchronization	Independent document updates across repositories	Continuous synchronization through traceability graph
Traceability Maintenance	Manual relationship verification	Automated lifecycle linkage generation
Document Preparation	Template completion after engineering activities	Continuous draft generation during development
Consistency Verification	Review-dependent	Rule-based validation before document synthesis
Change Impact Analysis	Manual assessment of affected documents	Graph traversal identifies dependent artifacts

Cross-Repository Integration	Repository-specific exports	Unified normalized data model
Regulatory Readiness	Final review performed near submission	Continuous documentation preparedness
Audit Support	Document-centric review	Artifact-centric provenance tracking

Table 3. Regulatory Framework Coverage by the Proposed Documentation Generation Framework

Framework	**Primary Focus**	**Supporting Artifacts**	**Framework Contribution**
IVDR Annex II	Technical Documentation	Requirements, Verification, Risk Files	Technical file generation
IVDR Annex III	Post-Market Documentation	Risk Records, Change History	Lifecycle maintenance documentation
FDA eSTAR	Electronic Submission	Software Description, SBOM, Test Reports	Structured submission generation
IEC 62304	Software Lifecycle	Requirements, Design, Verification	Lifecycle traceability
ISO 14971	Risk Management	Hazards, Controls, Verification	Risk documentation integration
Cybersecurity Guidance	Software Security Evidence	SBOM, Vulnerability Records	Cybersecurity documentation

Scientific evidence and domain knowledge required for summaries of clinical evaluation, benefit-risk analysis, regulatory rationale, intended use, and performance claims cannot be derived from software lifecycle artifacts alone. Automated document generation should therefore be seen as an augmentation technology that supports regulatory specialists, rather than as a substitute for regulatory review.

The effectiveness of the automated traceability further depends on the engineering discipline throughout software development. Incomplete metadata, inconsistent naming conventions, poorly maintained requirements or fragmented repository structures all degrade the quality of the graph construction and hence of the completeness of the documentation. Automated documentation helps organizations standardize engineering governance and discipline in managing the lifecycle. Natural Language Processing techniques for inferring se-mantic linkages add an additional dimension of possible ambiguous relationships between engineering artifacts. Hybrid approaches that combine the use of identifiers with semantic similarity substantially improve coverage, but inferred relationships need to be validated before they can be used as regulatory evidence. Therefore, rule-based confidence thresholds and expert approval mechanisms are still crucial components for the generation of trustworthy documentation.

Another key consideration is the regulatory acceptance of partially automated documentation. Current international regulations are based on evidence being documented, traceability through the life cycle, and reproducibility, rather than prescribing how documentation should be prepared. Nevertheless, regulators may demand transparency of automated generation processes, validation procedures, and provenance tracking. The proposed framework addresses this consideration by explicitly preserving the mappings between generated document sections and originating engineering artifacts, thus supporting auditability and regulatory inspection.

9. Conclusion and Future Work

Regulatory documentation is one of the most resource-consuming activities in medical device software development, owing to extensive traceability requirements spanning requirements engineering, software implementation, verification, validation, cybersecurity, configuration management, and risk management. Traditional documentation practices are often based on manual consolidation of engineering evidence from heterogeneous repositories, which increases the likelihood of inconsistencies and documentation maintenance issues.

In this paper, we have proposed a method for automated regulatory documentation generation based on structured software engineering artifacts. The proposed architecture offers a unified documentation pipeline that integrates artifact extraction, normalization, graph-based traceability construction, template-driven document synthesis and rule-based consistency validation to meet IVDR, FDA software submissions and IEC 62304 lifecycle expectations. The traceability graph is a flexible knowledge representation that links requirements, software modules, verification evidence, cybersecurity artifacts, Software Bills of Materials, configuration information and risk management documentation. Continuous synchronization allows documentation to evolve with software development, while maintaining artifact provenance and supporting regulatory auditability.

There are a number of promising extensions for future research. The integration of Large Language Models into controlled human-in-the-loop workflows could improve the generation of narrative regulatory sections with deterministic traceability. Knowledge graph reasoning can help improve automated change impact analysis in complex software ecosystems. Additional work could also expand framework support for quality management documentation (ISO 13485), cybersecurity compliance frameworks, the European Union Artificial Intelligence Act, Software as a Medical Device (SaMD) guidance, and post-market surveillance automation.

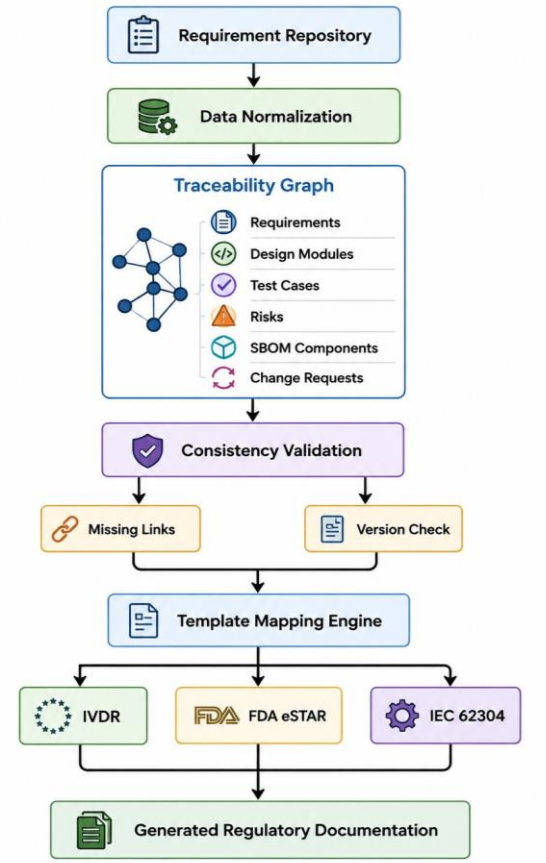

Figure 2. Traceability Graph Construction and Automated Regulatory Document Generation Workflow

References

[1] European Parliament and Council, "Regulation (EU) 2017/746 on In Vitro Diagnostic Medical Devices (IVDR)," Official Journal of the European Union, 2017.

[2] International Electrotechnical Commission, *IEC 62304: Medical Device Software—Software Life Cycle Processes*, Geneva, Switzerland, 2006 (Amendment 1:2015).

[3] International Organization for Standardization, *ISO 14971: Medical Devices—Application of Risk Management to Medical Devices*, Geneva, Switzerland, 2019.

[4] U.S. Food and Drug Administration, *21 CFR Part 820—Quality System Regulation*, Silver Spring, MD, USA.U.S. Food and Drug Administration, "Electronic Submission Template and Resource (eSTAR)," 2023.

[5] National Telecommunications and Information Administration, "The Minimum Elements for a Software Bill of Materials (SBOM)," 2021.

[6] OWASP Foundation, “CycloneDX Software Bill of Materials Specifica-tion,” Version 1.6, 2024.
[7] Linux Foundation, “Software Package Data Exchange (SPDX) Specifi-cation,” Version 3.0, 2024.
[8] Jama Software, “Requirements Traceability Best Practices for Regulated Industries,” White Paper, 2022.
[9] IBM Corporation, “IBM Engineering Requirements Management DOORS Next Documentation,” 2024.
[10] Siemens Digital Industries Software, “Polarion Application Lifecycle Management Documentation,” 2024.
[11] O. Gotel and A. Finkelstein, “An Analysis of the Requirements Trace-ability Problem,” *Proceedings of the First International Conference on Requirements Engineering*, Colorado Springs, CO, USA, pp. 94–101, 1994.