I.J. Information Technology and Computer Science, **2026, 3, 96-108**
Published Online on June 8, 2026 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2026.03.07

Context-oriented Framework for Determining Requirements Change Documentation Approaches

Denys Gobov*
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, 03056, Ukraine
E-mail: d.gobov@kpi.ua
ORCID iD: https://orcid.org/0000-0001-9964-0339
*Corresponding author

Oleksandra Zuieva
CS ltd, Kharkiv, 61072, Ukraine
E-mail: olekszueva@gmail.com
ORCID iD: https://orcid.org/0000-0001-9661-9657

Viktoriia Shashko
Donbass State Engineering Academy, Kramatorsk, 84313, Ukraine
E-mail: Viktoriia.Shashko@ddma.edu.ua
ORCID iD: https://orcid.org/0000-0002-6567-9873

Received: 14 February 2026; Revised: 16 March 2026; Accepted: 20 April 2026; Published: 08 June 2026

Abstract: Requirements change management is one of the core activities of a business analyst, directly affecting change impact analysis, stakeholder communication, and long-term system maintainability. While research on this topic examines in detail change processes, tracking methods, and change type classification, the problem of systematically documenting requirements changes remains underexplored. Existing research lacks a unified classification of change documentation approaches and context-sensitive recommendations for their selection, which limits their effectiveness in managing requirements.

To address this gap, this study develops a context-oriented framework for selecting approaches to requirements change documentation. The framework integrates three components: a conceptual model based on the Baseline–Delta–Target State triad, a taxonomy of documentation approaches, and a context-driven selection mechanism grounded in empirical evidence. A systematic literature review was combined with an analysis of the survey of 324 practicing business analysts from Ukrainian and international companies. Statistically significant associations between selected project context attributes and documentation practices were identified using the Chi-square test of independence and Cramer's V, while additional dimensions were supported through evidence from the literature.

The framework incorporates six documentation approaches: Full Target State, Delta-only, Target-driven Delta, Delta-driven Target, Parallel Use, and Hybrid Cycle. Four contextual dimensions emerge as key factors: project, environment, resources, and stakeholders. To support context-based selection of the change documentation approach, a matrix was developed that integrates the identified dependencies.

The results position requirements change documentation as a context-sensitive knowledge management mechanism rather than a universal procedural standard.

Index Terms: Requirements Change Management, Requirements Specification, Business Analysis.

1. Introduction

Requirements change is an inevitable part of any IT project, and requirements change management (RCM) has been recognized as a critical factor in successful software development. Changes can be caused by evolving business needs or laws, technological constraints, stakeholder feedback, or, more broadly, socio-economic and demographic factors. For business analysts in particular, this challenge is not limited to analyzing and handling change requests; it also concerns

This work is open access and licensed under the Creative Commons CC BY 4.0 License.

the way such changes are documented, communicated, and integrated into the existing requirements architecture.

In practice, inappropriate documentation may reduce the reliability of impact analysis, increase development costs, and, especially, increase the cost of long-term system maintenance. Therefore, the problem of requirements change documentation is of both theoretical and practical significance, especially in projects characterized by volatility, long development and operation periods, and high interdependence among systems and actors.

However, current studies do not provide a unified classification of documentation approaches and a structured method for selecting one based on the project context.

This gap is consequential for practicing business analysts and requirements engineers, who are responsible for selecting the information management approach during planning. The choice of approach affects not only the time required to prepare requirements and process change requests, but also the team's convenience in developing and supporting the product, especially for long-term projects in a distributed team environment. Consequently, a more explicit, context-oriented classification is needed to support both clarity and practical decision-making.

This study aims to develop a context-oriented framework for determining approaches to requirements change documentation and to identify the contextual factors that govern their applicability in information technology projects. The proposed framework integrates three components: (1) a conceptual model defining the relationships between the Baseline, Delta, and Target State; (2) a taxonomy of six documentation approaches derived from this model; and (3) a context-driven selection matrix that links project context profiles to recommended approaches.

2. Related Works

Research on RCM has concentrated on requirements engineering, software development, and change management, but the question "how should requirements changes be specified?" remains unanswered. Existing work has examined change management processes, traceability, and contextual challenges, but a framework for documenting and representing requirements changes in IT projects has not yet been analyzed. This distinction matters because documentation approaches influence not only requirement consistency but also communication, impact analysis, and solution maintainability [1].

A first major strand of the literature treats requirements changes as explicit objects of specification and classification rather than as informal updates to existing documentation. Research [2] proposes a structured approach in which software requirement changes are described through a dedicated change specification and then classified using attributes such as type, location, purpose, and focus. The key contribution of this approach is that change is treated as an artifact with its own descriptive structure, rather than being left implicit in the revised requirement text. Article [3] extends this logic in agile contexts by showing that requirements changes can be classified along multiple facets, including their types, sources, reasons, carriers, and triggering events. Research [4] advances the classification task by creating an ontology-based approach that distinguishes functional and technical changes and organizes them into more structured conceptual categories. Taken together, these studies indicate that one important response to requirements volatility is to make changes more explicit, semantically bounded, and systematically classifiable [2, 4].

However, the literature also shows that an artifact-centered view alone is insufficient. A second strand emphasizes process and governance frameworks for RCM. In this view, documenting a change is only one part of a broader managed lifecycle that includes identification, analysis, negotiation, approval, implementation, verification, updating of deliverables, and communication. Research [5] integrates RCM into a standards-oriented process structure rather than treating it as an ad hoc project practice. The quality of a change specification depends not only on the clarity of the change but also on how it is handled within governance procedures and organizational structures. Article [6] argues that successful RCM in global software development depends on organizational readiness, whereas research [7] considers change management capability as a staged maturity problem. These contributions are less informative about the internal form of the change record itself, but they show that documentation approaches are constrained by organizational capability, team distribution, and governance maturity [5, 6, 7]. A lightweight change notation may work poorly in an environment without shared practices, while a richer, semi-formal method may be impractical without process discipline.

A third strand considers the influence of requirements traceability and argues that requirement specification is incomplete unless it includes information on the origin and downstream relationships of requirements. Research [8] shows that linking requirements to their sources, decisions, and stakeholder inputs can improve understanding and support future maintenance. Study [9] reinforces this perspective through a systematic review of pre-requirements traceability, indicating that origin and rationale links are beneficial but also fragmented in practice. A more recent operational perspective [10] combines semantic analysis, trace links, and support for change impact and prioritization in agile projects. This strand is significant because it reveals that requirement change documentation can be oriented not only to the current target state or to the immediate delta, but also to the history of how and why the requirement changed. However, this approach has a limitation. Traceability studies evaluate the benefits of pre-requirements relationships not in the context of ongoing requirements change management, but rather primarily retrospectively (after solution development is complete) or experimentally (under controlled conditions). As a result, they do not provide a clear understanding of how to select a traceability approach within project constraints.

Thus, some approaches are final-state oriented, meaning that the primary documentation artifact remains the updated requirements specification, and the change is visible only in the resulting state. Final-state documentation remains

common in the IT industry, where natural language and templates dominate documentation practices [11]. As a result, the change is often hidden in revised artifacts rather than specified separately. In contrast, artifact-oriented methods, such as those in [2, 3], are more explicitly delta-oriented, as they highlight the change itself as the unit being described and classified. The traceability-oriented approach in [8, 9] is closer to history-oriented logic. The hybrid approach proposed in [10] includes structured change request records and traceability links. Crucially, none of these studies treats the approach to documenting requirements changes as a decision dependent on the project context. In [2, 3], delta-oriented documentation is considered the preferred approach in an agile environment, without analyzing the risks of losing consistency at the overall solution level or the factors that influence them. In [11], the dominance of final-state-oriented approaches is noted, without explaining whether this reflects sound practice or rather an inertial approach.

Several studies examine the influence of the project context on approaches to conducting business analysis activities, particularly in change management and requirements documentation [12, 13]. Studies [6, 14, 15] show that Global Software Development (GSD) environments amplify challenges related to communication, dependency analysis, and change coordination. Research [16] examines the emphasis on process formalization in geographically dispersed environments. These studies make two important contributions. First, they show that the project context should be incorporated into the documentation approach. Second, they suggest that the costs and benefits of documentation are context-sensitive. In a distributed environment, business analysts must trade off the lower overhead of lightweight documentation against the higher risk of misunderstanding [6, 14, 15, 16]. In this case, documenting change becomes a mechanism for coordination rather than merely record-keeping.

The same point appears in a different form in work on quality requirements and uncertainty. Research [17] shows that managing changes to quality requirements involves challenges, so some categories of requirement change may demand more careful specification and evaluation than others. Research [18] adds that uncertainty is often under-addressed in RCM, particularly in complex, interdisciplinary environments. At the same time, ecosystem-oriented work suggests that requirements management in multi-actor settings cannot be understood solely through single-team or single-organization assumptions [19]. These studies support the idea that the "right" way to document changes depends on the context in which the project is delivered: team distribution, business analyst experience, requirements type, level of uncertainty, organizational standards, and project maturity. The literature, therefore, points toward a context-sensitive view, but it rarely translates this view into guidance for selecting documentation approaches.

The practical implications of the choice of change documentation approach are examined in studies that link change processing to the subsequent development, testing, and support of solutions. Study [20] indicates that the content and quality of change documentation significantly affect effort, as poorly structured changes may lead to hidden costs during maintenance and implementation. Study [21] also emphasizes the importance of traceability and standardization of change documentation for effective project execution. A poorly specified change may seem inexpensive at the time of creation, but it increases uncertainty, rework, or support costs later. More complete change documentation can help mitigate these risks, especially in projects with multiple stakeholders, strict compliance requirements, or long-lived systems.

In summary, the literature on requirements changes in IT projects is rich in partial solutions but lacks a unifying analytical classification of approaches to specifying change. It also shows that the documentation approach is highly contingent on project context, including agility, distribution, maturity, uncertainty, and traceability needs. Still, explicit context-based selection guidance has not been provided.

3. Challenges in Requirements Change Documentation

Effective RCM represents a critical factor in the success of software projects [1, 2]. However, the process remains vulnerable to several persistent challenges, the severity of which is closely associated with the project context, organizational scale, and the level of formalization of internal processes [7, 22].

Empirical studies indicate that one of the fundamental barriers to effective RCM is the lack of transparency and systematic traceability of requirements. In particular, the absence of documented decision rationale complicates the assessment of the cumulative impact of changes on the project scope [1, 14, 23].

The analysis of existing practices reveals several dominant limitations:

- Communication barriers. Insufficient coordination between stakeholders and development teams frequently leads to misunderstandings in the interpretation of requirements, thereby increasing requirements volatility. This challenge is particularly acute in distributed project environments (Global Software Development), where geographical and temporal distances hinder frequent information sharing [6, 17, 24, 25].
- High rework effort. Delayed or incomplete impact analysis leads to instability in the requirements baseline. At the same time, inadequate documentation of requirement changes increases the likelihood of using outdated artifacts, which triggers recursive cycles of rework [1, 5, 12, 26].
- Loss of project knowledge. The transition of project knowledge into an implicit form increases dependency on individual team members and raises the risk of knowledge loss. Furthermore, the absence of traceability to pre-requirement specification artifacts complicates the analysis of the rationale behind individual requirements and system verification when an up-to-date baseline is unavailable [1, 9, 27, 28].
- Insufficient consideration of non-functional requirements. Fragmented documentation of quality-related

requirements (NFRs) frequently leads to architectural inconsistencies. Empirical data suggest that NFRs are often underestimated or integrated into general documentation without proper evaluation of quality attributes, resulting in long-term technical debt [27, 29, 30].

In the study [22], which surveyed practicing business analysts, the most critical problems faced by specialists in software requirements development were identified. The issues related to requirements documentation are illustrated in Fig. 1.

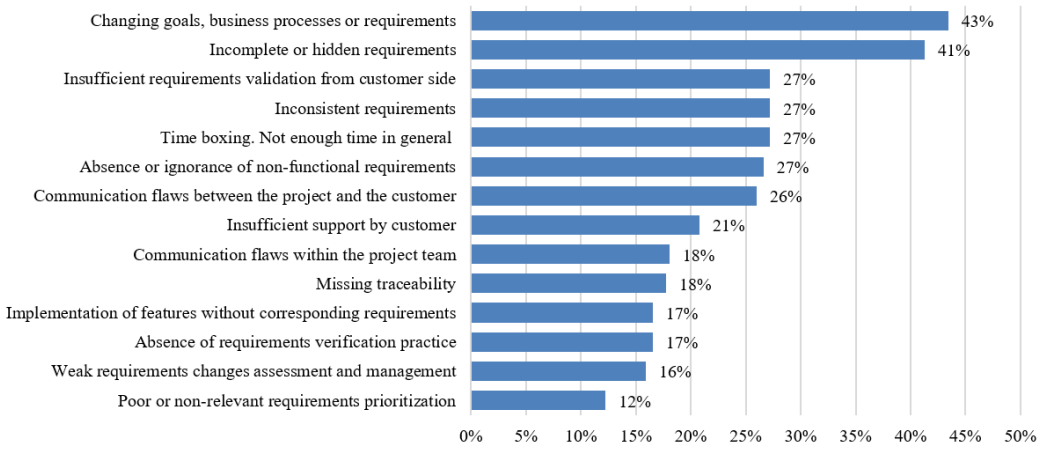


Fig. 1. The most critical problems with creating requirements

The identified issues suggest that many challenges traditionally associated with requirements engineering are significantly amplified by inadequate documentation practices, particularly in ensuring consistency, traceability, and completeness of requirements artifacts. This suggests that the availability of systematic and context-appropriate approaches to requirements documentation largely determines the effectiveness of RCM.

4. Requirements Change Documentation Approaches

This section presents the first two components of the proposed framework: the conceptual model, Baseline-Delta-Target State, and the taxonomy of requirements change documentation approaches.

The systematization of RCM requires a clear definition of the states of the requirements information model, in which the specification is considered a dynamic body of knowledge evolving from an initial to a target state. This evolution can be described through the triad of concepts: Baseline, Delta, and Target State.

Baseline (A) represents a set of verified and approved requirements that capture the system's state at a specific point in time [31, 32].

Delta (Δ) is defined as the specified difference between the current state of the system and the desired modification. In the scientific literature, the concept of a "directed delta" is also used to describe a sequence of transformation operations (addition, deletion, modification) which, when applied to version v1, produce version v2 [31, 32, 33].

Target State (B) is the state of the system resulting from the successful application of Delta to the existing Baseline ($A + \Delta = B$). This state reflects the fulfillment of business needs and the expected value delivered to the organization [32, 34].

The relationships between Baseline, Delta, and Target State are illustrated in Fig. 2.

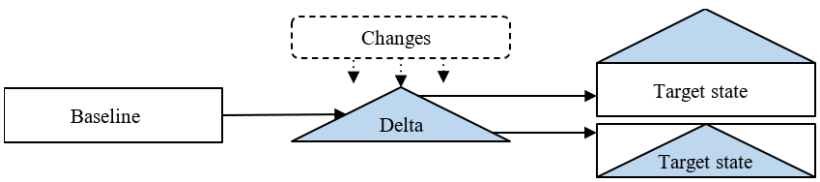


Fig. 2. The relationships between baseline, delta, and target state

The analysis of software configuration management (SCM) and requirements engineering (RE) practices indicates that different approaches to documenting requirements changes emerge as combinations of these paradigms. In this context, documentation approaches differ in how they represent system states, capture changes, and maintain the consistency of the requirements baseline.

Based on this conceptual foundation, approaches to requirements change documentation can be systematized into a set of distinct approaches that reflect different ways of balancing state representation and change tracking (Table 1).

Table 1. Systematization of approaches to documenting change requirements

Approach	Historical context	Essence and logic	Advantages	Disadvantages	Artifact
Full Target State	It was formed in the 1970s and 1980s within the framework of classical SCM systems (e.g., RCS) and the waterfall development model. It is defined in the IEEE 830-1984 standard [31, 35, 36]	Based on preserving the integrity of the system states. Each change is considered a transition of the system from one valid state to another and requires a complete review and update of the basic specification.	High transparency and relevance; no need to reconstruct history to understand the current logic of the system; no duplication.	Extremely high labor intensity with frequent changes. Difficult to adapt to unpredictable changes [37]	SRS / FSD(Use Case)
Target-driven Delta	The technical foundation of the approach was laid in early version control systems such as SCCS (1975) and DSEE (1985). [31,38] The approach has been popularized in methodologies such as SAP Fit-to-Standard [39]	The main artifact is the Baseline. Instead of storing separate copies of files, all versions were stored in a single dataset, where fragments of text were highlighted with special control tags or attributes; i.e., the delta is only a temporary reference to specific changed points.	Saving resources on the description of the delta, preserving the integrity of the Baseline.	It is difficult for the development team to identify the full scope of work.	Use Case
Delta-driven Target	It was formed in the late 1970s - 1990s as an alternative to traditional models within the concept of intentional versioning. Implemented in the PIE systems (1980), developed at XEROX PARC for Smalltalk programs, and DaSC (1995) [31]	The target state is not stored as a document, and the system is described as a set of successive deltas (change sets/layers), whose sum forms the current image of the product.	Convenience for incremental development; effective management of product variability; preservation of the history of architectural solutions	Difficulty understanding the overall architecture for new participants. The difficulty of reconstructing a holistic image increases with project duration.	User story + acceptance criteria
Delta-only	2000s. Spread after the publication of the Agile Manifesto [40] and the development of task trackers [26, 35]	Only the content of the change is recorded without updating the specification	Maximum speed of response to changes and delivery rate; minimum costs for initial document creation.	Loss of context; risk of contradictions when accumulating deltas.	User story + acceptance criteria
Parallel Use	2010s. Development within Scaled Agile (SAFe) and Globally Distributed Development (GSD). [13, 29]	The change is described as a separate request and entered into the target document simultaneously.	Best traceability for critical systems; eliminates communication gaps in distributed teams.	High information redundancy; risk of desynchronization between delta and target description	RTM, Use Case+User story + acceptance criteria
Hybrid Cycle	Began to be actively researched in recent years (2020s) as an answer to the problem of documentation degradation in scalable Agile projects [7, 41]	Using deltas for ongoing work with periodic full baseline updates at defined "checkpoints."	Optimal balance between Agile speed and documentation quality; managed degradation risk [10, 42]	Requires high team discipline and regular documentation reviews.	User story + acceptance criteria+Use Case (SRS)

The six approaches define a space between two extremes – maximum specification integrity (full target state) and maximum rate of change (delta changes only) – with intermediate approaches resolving this tension through various tradeoffs in traceability, resource efficiency, and maintainability. Each approach historically emerged as a response to the limitations of its predecessor; none is universally applicable, and the framework presented below explores how to reconcile them within the project context.

5. Empirical Analysis of Documentation Practices

To determine the current state of approaches to requirements documentation and management, a survey was conducted among practitioners [43]. Three hundred twenty-four business analysts from Ukrainian and international companies participated. The following question regarding managing changes in requirements was asked:

"How do you deal with changing solution requirements after the requirements are signed off (formally approved)?":

- We do not update the solution requirements specification once formally approved. We create a separate document/backlog item for the change (Approach 1).
- We create a separate document/backlog item and establish reciprocal links between the new and old document/item. (Approach 2).
- We update our requirements specification even after formal sign-off. Then we reiterate the sign-off (acceptance) procedure with respect to the changes. (Approach 3).

- 314 respondents chose one of the options mentioned above; 10 participants selected the "Other" option.

According to the survey results shown in Fig. 3, respondents more often update the requirements specification after formal sign-off (48%).

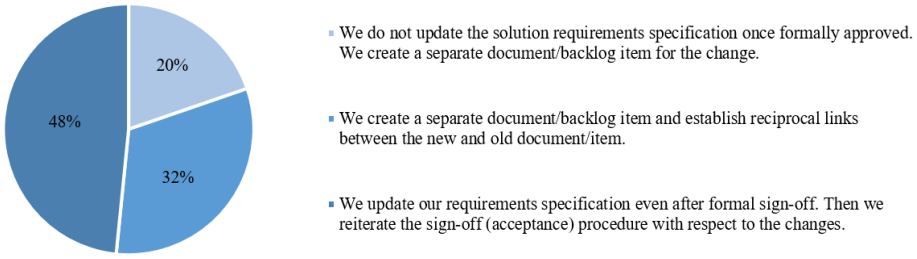


Fig. 3. The survey results on dealing with changing solution requirements

To examine the relationship between project context and the approach to managing requirements changes, the Chi-square test of independence was applied at the 0.05 significance level. Since the Chi-square test has known limitations when applied to contingency tables of varying dimensions, Cramer's V, an association measure derived from the Chi-square statistic, was calculated to assess the strength of association.

The strongest association (p-value = 0.003, Cramer's V = 0.175) was observed for the project context attribute "way of working". Respondents were able to select the following ways of working: Agile/Lean; rather Agile; hybrid; rather plan-driven; and plan-driven. The distribution is given in Table 2.

Table 2. An association with the way of working and the requirements documentation approach

Way of Working	Approach 1	Approach 2	Approach 3	Overall
Agile/Lean	17.90%	44.8%↑	37.30%	67
Rather agile	11.90%	33.90%	54.2%↑	118
Hybrid	21.00%	30.90%	48.10%	81
Rather plan-driven	40.5%↑	9.5%↓	50.00%	42
Plan-driven	33.30%	16.70%	50.00%	6
Overall	19.70%	31.80%	48.40%	314

Business analysts working in a "plan-driven" or "rather plan-driven" environment more often create a separate document/backlog item for the change. Separate documents with established reciprocal links are used in Agile/Lean projects. Projects with a "rather agile" approach are more likely to update the requirements specification even after formal sign-off or create a separate document with established reciprocal links.

An association with company type was also observed (p-value = 0.0458, Cramer's V = 0.1045). The distribution is given in Table 3.

Table 3. An association with the company type and requirements documentation approach

Company type	Approach 1	Approach 2	Approach 3	Overall
IT Outstaff	15.10%	**37.7%** ↑	47.20%	53
IT Outsource	19.00%	**38.6%** ↑	42.50%	153
IT Product	23.20%	**16.1%** ↓	**60.7%** ↑	56
Non-IT	23.10%	23.10%	53.80%	52
Overall	19.70%	31.80%	48.40%	314

Both associations confirm that documentation practice is not independent of project context, and the stronger of the two – methodology – provides meaningful, if partial, empirical grounding for the framework.

IT Product companies are the only significant outlier (standardized residual of −2.09): only 16.1% use Approach 2, compared with the expected 32%. In contrast, the dominance of Approach 3 (60.7%) reflects a "single source of truth" culture with full re-approval for any changes.

Both IT Outstaff and IT Outsource companies score above average on Approach 2 (~38% vs 31.8% in the sample).

Non-IT has a profile close to the average, with a slight bias towards Approach 3 (53.8%), reflecting a corporate culture of formal approvals.

6. Factors and Framework for Requirements Change Documentation

This section presents the last component of the proposed framework: the context-driven selection mechanism. The analysis of statistically significant dependencies between project parameters and the content of business analytics artifacts [30] allows us to assert that the choice between approaches for documenting changes (Delta) and updating the holistic model (Baseline/Target State) is not arbitrary, but is dictated by the systemic constraints of the environment. To systematize this influence, four main dimensions are identified that create specific constraints for information management.

Project Dimension

This dimension covers the project's internal characteristics and constraints that determine the rationality of costs supporting the Baseline.

- Lifecycle duration and phasing: Long-term projects require incremental fixation of the Baseline to prevent knowledge degradation [8, 9]. The value and structure of documentation evolve according to the phases of the SDLC [15, 24]:
- Inception / Elaboration: Building a knowledge base. The main artifact is the Target State, as the absence of an implemented Baseline makes the description of Delta impractical [2]. The main goal is to ensure a common understanding of the future state by all stakeholders [3].
- Construction: A phase of high change amount, in which delivery speed (Delta-only) competes with system description integrity (Full Target State). In the Agile context, high team maturity supports the Hybrid Cycle [44], while low maturity often leads to a regression of documentation to unstructured deltas [3].
- Transition / Maintenance: Stabilization phase. Target State documentation is minimized in favor of a reference system (Target-driven Delta), which maintains the system's operational capability with minimal business analysis costs [22].
- Change Ratio: It has been established that, with 80% functionality stability (Legacy system), a complete update of the Target State is economically irrational, which justifies the use of the Target-driven Delta approach [2]. However, with a radical transformation (>70% of changes), maintaining the old Baseline becomes critically difficult ("maintenance nightmare") [1, 22], which forces analysts to prefer autonomous Delta descriptions.
- Project Scale: As the scale of a project increases, so does the likelihood of cascading consequences, which require detailed impact analysis that cannot be conducted without a confirmed Target State [19, 22].
- Organizational type and methodology: Product companies tend to maintain a Target State more often due to their long asset ownership cycles. In contrast, fixed-budget outsourcing projects are more likely to use approaches that lean toward partial documentation, such as Delta-driven Target, where documentation accumulates incremental modifications by the time the contract is completed [3, 31]. Agile methodology is a powerful driver of the Delta-only approach, given its priority of working software over comprehensive documentation.

A. Environmental Dimension

This dimension captures external determinants that are not directly controllable by the project team and require primarily reactive adaptation in decision-making:

- The degree of environmental uncertainty: The high volatility of requirements, typical of innovative projects (R&D) and startups, stems from an initial knowledge deficit, with more than 50% of requirements having a high probability of changing over the life cycle [18]. Under such conditions, a detailed description of the Target State becomes redundant and economically irrational, since the level of uncertainty directly affects the choice of development approach, favoring Agile methods [5]. The Delta-only approach minimizes the sunk cost of updating large amounts of documentation that may become obsolete in the next iteration [22].
- Regulatory constraints: Industry standards (ISO, HIPAA, GDPR) preclude reliance on deltas alone [14, 45]. Compliance standards create a foundation of constraints that cannot be ignored, as they require constant updating of the Target State to ensure legal compliance, audit transparency, and traceability from regulatory norms to specific requirements, and make pure Delta-only approaches impractical in most regulated contexts [8]. Failure to comply with these requirements can lead to legal conflicts and a loss of the organization's reputation [45].
- Safety requirements and risk tolerance: The literature reviewed consistently notes that safety-critical systems with low risk tolerances require detailed target state documentation [9, 11]; no contrary cases have been reported, although comparative empirical studies of this relationship remain limited. It is due to the need to conduct a preventive impact analysis of technical changes on safety requirements before their implementation [44]. For such systems, the creation of Change Management Boards is mandatory to ensure the quality, safety, and stability of each product version [29]. Formal specifications and verified baselines prevent architectural degradation, which is critical for the viability of such systems [11, 19].

B. Team & Resource Dimension

This dimension reflects a team's ability to maintain a particular knowledge architecture, which is directly dependent on human capital and available assets.

- Analysts' professional maturity: Empirical evidence confirms that the level of expertise of the specialist directly correlates with the type of specification artifacts chosen [11, 30]. Experienced analysts are statistically more likely to choose modeling and complex text structures, considering documentation as a strategic asset to ensure traceability and architectural integrity. In contrast, novice analysts (with up to 5 years of experience) tend to view documentation as an operational task and prefer the Delta approach. High team maturity enables the maintenance of complex hybrid models, ensuring the long-term stability of knowledge.
- Resource constraints (time, staff, budget): Time constraints or limited availability of business analysts are among the main barriers to implementing a system-wide requirements management process [15, 24]. In resource-constrained environments, Target State support is often perceived as a secondary process because it does not create immediate value for ongoing development [9]. It forces the team to use a minimalist approach (Delta-only), which saves resources during the specification stage but increases the risk of accumulating documentation debt and incurring significant rework costs in the future [24, 46].
- Geographical Distribution (GSD): Projects developed by distributed teams face additional challenges: different time zones, cultural barriers, and the lack of face-to-face interaction lead to communication problems. In such settings, brief descriptions of changes are insufficient to maintain a coherent picture among stakeholders [6, 42]. To prevent knowledge loss and misinterpretation, distributed teams require information redundancy and the use of common platforms for documentation, consistent with the Parallel Use approach. Under such conditions, each change must be reflected in the Target State to provide a single source of information across all locations [7, 25].

C. Stakeholder Dimension

This dimension defines an external request for the form of information presentation, driven by the need to synchronize the product vision among all participants in the process.

- Customer Involvement: A high level of stakeholder involvement is a critical success factor for a project and directly correlates with the need for a detailed Target State to verify business objectives [42]. The customer needs to see the resulting solution to ensure alignment with the business strategy. In contrast, the development team often needs only an isolated "delta" (in the form of User Stories or Tasks) for immediate implementation [3, 44]. According to an analysis of empirical research from the last decade (2013–2021), a lack of communication and coordination is identified as a critical barrier in 45% of cases, leading directly to an increase in change requests due to initial ambiguity in requirements [17, 27]. The Parallel Use or Hybrid approaches, which combine the flexibility of iterations with the support of a holistic specification of the system's target state, are recommended for projects with a high level of stakeholder interaction.
- Documentation Use Objectives: The dependencies identified in [30] indicate a direct relationship between the purpose of the document and the choice between Delta and Baseline:
- For user training, operational software support, and compliance audits, the Target State is preferred because it provides a complete understanding of the system's functionality [8, 11].
- For cost and effort estimation or rapid task setting in Scrum teams, a Delta-based approach that focuses on specific modifications is more effective [23, 44].
- When using documentation as a strategic asset for long-term product ownership, companies tend to maintain an up-to-date Baseline, while in outsourcing projects with limited budgets, regression to fragmented deltas is more common.

The lack of a single standard for documenting changes often becomes a barrier to effective interaction, so adapting the form of requirements presentation to a specific group of stakeholders (Decision-makers vs. Developers) is a necessary condition for project manageability.

Based on integrating the conceptual model, the six-approach taxonomy, and the contextual dimensions, the context-oriented framework is defined through a selection matrix (Table 4). The matrix enables practitioners to map a given project's contextual profile to the most appropriate documentation approach.

The analysis indicates that the form of documenting requirements changes and the architecture of project requirements are shaped by balancing the intensity of feature delivery with the need to maintain the integrity of architectural knowledge about the system. The selection of an appropriate documentation approach is determined by a combination of contextual factors, among which regulatory constraints, the availability of business analysis resources, the team's organizational structure, and the system lifecycle phase are key.

In domains with strict compliance and security requirements, maintaining a consistent Target State is necessary to control risk and ensure traceability of requirements. In such contexts, a comprehensive system specification functions as a stabilizing element of the knowledge base and ensures the manageability of complex requirements architecture.

Table 4. Matrix for defining requirements documentation approach depending on the project context

Factor / Approach	Full Target State	Delta-only	Parallel Use	Target-driven Delta	Delta-driven Target	Hybrid Cycle
Regulatory constraints & Safety requirements	High	Low	High	Middle	Low	Middle
BA resource constraints	High	Low	High	Middle	Middle	Middle / High
The degree of uncertainty	Low	High	Middle	Low	High	Middle
Geographical Distribution	Middle	Low	High	Middle	Low	Middle
Project duration	Long-term	Short-term	Long-term	Long-term	Medium / Short-term	Long-term
SDLC phase	Inception / Elaboration	Construction	The entire cycle	Maintenance	Construction / Maintenance	Construction
Project Scale	Big	Small	Big	Any	Middle	Big
Customer Involvement	High	Low	High	Middle	Low	High
Change Ratio	< 20% (Legacy)	> 70%	Any	< 20% (Support)	> 50% (Active)	Any
Analysts' professional maturity	High	Low	High	Middle	Middle	High
Organizational type	Product	Startup	GSD-Enterprise	Product	Outsourcing	Any
Methodology	Waterfall	Agile	Unified Process	Unified Process	Agile	Hybrid
Documentation Use Objectives	Knowledge preservation and compliance assurance	Rapid change communication and task execution	Alignment between development and compliance needs	Efficient system maintenance with controlled evolution	Incremental knowledge accumulation under constraints	Balancing adaptability and knowledge integrity

Conversely, in projects with significant time or resource constraints, incremental approaches to documenting changes (Delta-only) tend to dominate, as they enable a high pace of iteration. However, prolonged reliance on purely incremental documentation leads to fragmentation of system knowledge and gradual loss of a coherent understanding of system logic.

In more complex organizational contexts, hybrid documentation models emerge. For globally distributed teams, a parallel approach of incremental changes and maintaining an up-to-date Target State is typical. During the maintenance phase, the Target-driven Delta approach is widely used, in which changes are recorded as incremental modifications linked to the existing Baseline. In outsourcing projects, a Delta-driven Target model is often observed, in which documentation evolves through the accumulation of changes and is consolidated only at key stages of system development. For teams with high Agile maturity, the Hybrid Cycle model is characteristic, combining incremental documentation with periodic revision of the system baseline.

Thus, requirements change documentation approaches can be considered adaptive mechanisms for managing system knowledge, balancing the speed of iteration (Delta) and the manageability of the knowledge base (Baseline). The effectiveness of a selected approach depends on the analyst's ability to identify the project's dominant contextual factors: in highly regulated environments, priority is given to maintaining a consistent Target State, whereas in contexts with high market uncertainty, incremental Delta-based approaches prevail.

A critical aspect of documentation management is the timely identification of the so-called degradation point – the moment at which the accumulated volume of incremental changes renders the current Baseline obsolete. Transitioning to consolidation or full revision of the Target State at this stage helps prevent degradation of architectural knowledge and ensures the manageability of complex software systems.

Modern RCM increasingly relies on intelligent methods to overcome the limitations of fragmented documentation. As demonstrated in [47], ensuring requirements quality and, consequently, an effective development process requires the construction of a requirements architecture that supports traceability, completeness, and consistency. Achieving these quality criteria is particularly challenging under the Delta-only approach, as documenting only incremental changes without updating a comprehensive specification increases the risk of context loss.

7. Conclusions

The conducted literature review demonstrates that existing research in requirements engineering primarily focuses on change management processes, traceability, and the classification of change types. In contrast, the problem of systematically selecting approaches to requirements change documentation remains insufficiently formalized. It indicates a research gap: the lack of a unified perspective on documentation as a mechanism for managing knowledge of system evolution.

The analysis of practical challenges faced by business analysts shows that key issues include loss of knowledge integrity, fragmentation of change-related information, and increasing complexity of maintaining a consistent requirements baseline. These challenges are largely driven by the absence of context-aware criteria for selecting documentation approaches.

The study systematizes methods for documenting requirements changes. This study demonstrates that requirements change documentation cannot be reduced to a single universal approach but should be considered as an adaptive

mechanism for managing architectural knowledge under varying project conditions.

The identification of contextual factors across four dimensions – Project Dimension, Environmental Dimension, Team & Resource Dimension, and Stakeholder Dimension – shows that the choice of documentation approach is determined by project context. Different combinations of these factors impose distinct requirements on the structure and completeness of documentation artifacts.

The results of the empirical survey confirm a relationship between project context and applied documentation approaches: regulatory environments favor the Target State-oriented approach, whereas resource-constrained contexts favor the incremental Delta-based approach.

Based on integrating theoretical and empirical results, a context-strategy matrix was developed to formalize the selection of documentation approaches. The proposed classification identifies six approaches and links them with contextual factors, offering a structured guideline for documentation decisions and a starting point for context-sensitive practice. Its practical utility is conditional on validation in settings beyond the Ukrainian IT sector from which the survey data were drawn.

As a result, the context-oriented framework was developed, comprising three integrated components: the Baseline-Delta-Target State conceptual model, the six-approach taxonomy, and the context-strategy selection matrix.

The study results are limited by two factors. First, only practicing specialists from Ukraine participated in the survey. Second, statistically significant effects were found for two project context factors (methodology and company type); the remaining factors were identified through a literature review, including empirical studies and surveys conducted across various cultural and subject contexts. Future work should address each of these gaps through multi-country replication, observational studies, and expert validation of the taxonomy. Besides that, future research may explore automated support for documentation processes, including intelligent methods for traceability, change-impact analysis, and detecting knowledge degradation in complex systems. Advances in artificial intelligence technologies such as neural networks with self-attention mechanisms [48] and LLM-based impact analysis [49, 50] may impact the identified factors, particularly given the reduced effort required to update requirements, which will require a re-evaluation of the matrix as these tools mature and become more widely adopted by project teams.

All the Declarations and Statements

Author Contributions Statement

D. Gobov: Conceptualization, Methodology, Investigation: Proposed the core research idea and defined the study objectives. Developed the Baseline–Delta–Target State conceptual triad and the unified taxonomy of six requirements change documentation approaches. Designed the survey for empirical data collection. Developed the context-driven selection matrix, integrating the framework components. Conducted statistical analysis using the Chi-square test and Cramer's V. Writing – Review & Editing: Reviewed and edited the manuscript.

O. Zuieva: Data Curation, Formal Analysis, Validation: Conducted empirical analysis of documentation practices and systematized existing approaches. Participated in survey data collection and interpretation. Contributed to the identification and classification of contextual factors. Validated the framework components against empirical evidence. Writing – Review & Editing: Reviewed and edited the manuscript, ensuring clarity and coherence.

V. Shashko: Writing – Original Draft, Validation: Conducted the systematic literature review and contributed to identifying the research gap. Drafted the manuscript and structured the theoretical background. Participated in the validation of the proposed framework and preparation of the manuscript for submission.

All authors have read and agreed to the published version of the manuscript.

Conflict of Interest Statement

The Authors declared no conflicts of interest.

Funding Declaration

This research received no external funding.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Ethical Declarations

This study did not involve human participants or animals.

Acknowledgements

We sincerely thank the experts for their professional evaluation and valuable recommendations, which have contributed to improving the quality of the experiment and the reliability of its results.

Declaration of Generative AI in Scholarly Writing

During the preparation of the manuscript, the authors carefully reviewed the content and take full responsibility for the accuracy and integrity of the manuscript.

Abbreviations

The following abbreviations are used in this manuscript:

RCM - Requirements Change Management
NFRs - Non-functional requirements
SCM - Software Configuration Management
RE - Requirements Engineering
SRS - Software Requirements Specification
FSD - Functional Specification Document
RCS - Revision Control System
RTM - Requirements Traceability Matrix
GSD - Global Software Development
AI - Artificial Intelligence
IT - Information Technology

APPENDIX

None

References

[1] Jayatilleke, S., & Lai, R. (2018). A systematic review of requirements change management. Information and Software Technology, 93, 163-185. DOI: 10.1016/j.infsof.2017.09.004

[2] Jayatilleke, S., Lai, R., & Reed, K. (2018). Managing Software Requirements Changes Through Change Specification and Classification. Computer Science and Information Systems, 15(2), 321-346. DOI: 10.2298/CSIS161130041J.

[3] Madampe, K., Hoda, R., & Grundy, J. (2022). A Faceted Taxonomy of Requirements Changes in Agile Contexts. IEEE Transactions on Software Engineering, 48(10), 3737-3752. DOI: 10.1109/TSE.2021.3104732

[4] Sakhrawi, Z., Sellami, A., & Bouassida, N. (2020). Requirements Change Requests Classification: An Ontology-Based Approach. International Conference on Intelligent Systems Design and Applications, 1181, 487–496. DOI: 10.1007/978-3-030-49342-4_47

[5] Anwer, S., Wen, L., Rout, T., & Wang, Z. (2018). Introducing requirements change management process into ISO/IEC 12207. In Software Process Improvement and Capability Determination (SPICE 2018) (CCIS, Vol. 918, pp. 185-199). Springer. DOI: 10.1007/978-3-030-00623-5_13

[6] Akbar, M. A., Mahmood, S., Huang, Z., Khan, A. A., & Shameem, M. (2020). Readiness model for requirements change management in global software development. Journal of Software: Evolution and Process, 32(10), e2264. DOI: 10.1002/smr.2264

[7] Akbar, M. A., Khan, A. A., Mahmood, S., & Mishra, A. (2023). SRCMIMM: The Software Requirements Change Management and Implementation Maturity Model in the Domain of Global Software Development Industry. Information Technology and Management, 24, 195-219. DOI: 10.1007/s10799-022-00364-w

[8] Krause, J., Kaufmann, A., Riehle, D., & Jung, M. (2022). The Benefits of Pre-Requirements Specification Traceability. IEEE 30th International Requirements Engineering Conference (RE), 166-177. DOI: 10.1109/RE54965.2022.00021

[9] Mucha, J., Kaufmann, A., & Riehle, D. (2024). A Systematic Literature Review of Pre-Requirements Specification Traceability. Requirements Engineering, 29, 119–141. DOI: 10.1007/s00766-023-00412-z

[10] Ilyas, I. et al. (2025). Optimized Change Management Process Through Semantic Requirements and Traceability Analysis Tool. Journal of Engineering, 2296387-2296404. DOI: 10.1155/je/2296387

[11] Franch, X., Palomares, C., Quer, C., Chatzipetrou, P., & Gorschek, T. (2023). The State-of-Practice in Requirements Specification: An Extended Interview Study at 12 Companies. Requirements Engineering, 28, 377-409. DOI: 10.1007/s00766-023-00399-7

[12] Gobov, D., & Sokolovskiy, N. (2025). An Association Rule Mining for Requirement Document Structure in IT Projects. Advances in Computer Science for Engineering and Education VII, 242. DOI: 10.1007/978-3-031-84228-3_27

[13] Gobov, D., & Zuieva, O. (2025). Software Quality Attributes in Requirements Engineering. International Journal of Information Technology and Computer Science (IJITCS), 17(4), 38-48. DOI: 10.5815/ijitcs.2025.04.04

[14] Anwer, S., Wen, L., & Wang, Z. (2019). A systematic approach for identifying requirement change management challenges: Preliminary results. In Proceedings of the 23rd International Conference on Evaluation and Assessment in Software Engineering (EASE 2019) (pp. 230-235). ACM. DOI: 10.1145/3319008.3319031

[15] Akbar, M. A., Khan, A. A., Khan, A. W., & Mahmood, S. (2020). Requirement change management challenges in GSD: An analytical hierarchy process approach. Journal of Software: Evolution and Process, 32(7), e2246. DOI: 10.1002/smr.2246

[16] Bhatti, S. N. et al. (2020). Optimized Approach in Requirements Change Management in Geographically Dispersed Environment. International Journal of Advanced Computer Science and Applications, 11(3), 520-525. DOI: 10.14569/IJACSA.2020.0110366

[17] Ahmad, J. et al. (2022). Quality Requirement Change Management's Challenges: An Exploratory Study Using SLR. IEEE Access, 10, 127575-127588. DOI: 10.1109/ACCESS.2022.3224593

[18] Gräßler, I., Pottebaum, J., Oleff, C., & Preuß, D. (2021). Handling of Explicit Uncertainty in Requirements Change Management. Proceedings of the Design Society, 1687–1696. DOI: 10.1017/pds.2021.430

[19] Malcher, P., Viana, D., Antonino, P. O., & dos Santos, R. P. (2025). Towards an Understanding of Requirements Management in Software Ecosystems. Information and Software Technology, 180, 107672. DOI: 10.1016/j.infsof.2025.107672
[20] Chua, B. B., & Verner, J. (2010). Examining Requirements Change Rework Effort: A Study. International Journal of Software Engineering & Applications, 1, 48–64. DOI: 10.5121/ijsea.2010.1304
[21] Khan, H. U. et al. (2021). Empirical Investigation of Critical Requirements Engineering Practices for Global Software Development. IEEE Access, 9. pp. 93593–93613, 2021, DOI: 10.1109/ACCESS.2021.3092679
[22] Gobov, D., & Titlova, O. (2023). Towards identifying challenges in business analysis on IT projects–a practical study. Radioelectronic and Computer Systems, (2), 193-206. DOI: 10.32620/reks.2023.2.16
[23] Koulecar, N., & Ghimire, B. (2024). Agile requirement change management model for global software development. arXiv preprint arXiv:2402.14595. DOI: 10.48550/arXiv.2402.14595
[24] Majeed, B., & Sabahat, N. (2024). Requirements change management (RCM) tool for Pakistan software industry. Pakistan Journal of Scientific Research, 3(2), 190-200. DOI: 10.57041/pjosr.v3i2.1018
[25] Wahbeh, N., Admodisastro, N., Hassan, S. et al. (2024). Requirement Change Management in Global Software Development: A Systematic Review. DOI: 10.21203/rs.3.rs-4253959/v1
[26] Ben Charrada, E. (2013). Supporting Requirements Update during Software Evolution. Dissertation, University of Zurich. DOI: 10.5167/uzh-89558
[27] Janes, A., Remencius, T., Sillitti, A., & Succi, G. (2013). Managing changes in requirements: an empirical investigation. Journal of Software: Evolution and Process, 25, 1273-1283. DOI: 10.1002/smr.1602
[28] Spijkman, T., Dalpiaz, F., & Brinkkemper, S. (2022). Back to the Roots: Linking User Stories to Requirements Elicitation Conversations. IEEE 30th International Requirements Engineering Conference (RE), 281-287. DOI: 10.1109/RE54965.2022.00042
[29] Satapathy, B., Sharma, A., Dash, M. et al. (2025). Driving Innovation Through Change: Release Management With a Focus on Nonfunctional Requirements and Emerging Technologies. Cureus Journal of Computer Science, 2. DOI: 10.7759/s44389-025-03354-5
[30] Gobov, D., & Zuieva, O. (2023). Identifying the dependencies between IT project context and business analysis document content. Innovative Technologies and Scientific Solutions for Industries, 2(24), 39–53. DOI: 10.30837/itssi.2023.24.039
[31] Conradi, R., & Westfechtel, B. (1998). Version models for software configuration management. ACM Computing Surveys, 30(2), 232–282. DOI: 10.1145/280277.280280
[32] International Institute of Business Analysis. (2015). A Guide to the Business Analysis Body of Knowledge (BABOK® Guide) (Version 3.0).
[33] Ya'u, B. I., Nordin, A., Salleh, N., & Mustapha, S. W. (2019). Requirement patterns: An approach for streamlining requirements engineering in software product families. International Journal of Advanced Trends in Computer Science and Engineering, 8(1.4), 507-514. DOI: 10.30534/ijatcse/2019/7981.42019
[34] Mehmood, W., Khan, A. W., Aslam, W., Ahmad, S., El-Sherbeeny, A. M., & Shafiq, M. (2022). Requirement design for software configuration and system modeling. Intelligent Automation & Soft Computing, 32(1), 441-454. DOI: 10.32604/iasc.2022.016116
[35] Ernst, N., Borgida, A., Jureta, I. J., & Mylopoulos, J. (2014). An Overview of Requirements Evolution. Evolving Software Systems. DOI: 10.1007/978-3-642-45398-4_1
[36] IEEE. (1984). IEEE Guide for Software Requirements Specifications. IEEE Std 830-1984, 1-26. DOI: 10.1109/IEEESTD.1984.119205
[37] Koegel, M., Herrmannsdoerfer, M., Li, Y., Helming, J., & David, J. (2010). Comparing state- and operation-based change tracking on models. Proceedings of the 14th IEEE International Enterprise Distributed Object Computing Conference (EDOC '10), 163–172.
[38] Rochkind, M. J. (1975). The Source Code Control System. IEEE Transactions on Software Engineering, SE-1(4), 364–370
[39] Aka, V. P. K., & Samuel, K. K. M. (2023). Predictive modeling for brownfield migration from SAP ECC 6.0 to S4HANA: A machine learning approach for effort evaluation under the SAP RISE program. International Journal of Computer Engineering and Technology, 14(1), 232-248. DOI: 10.34218/ijcet_14_01_018
[40] Fowler, M., & Highsmith, J. A. (2001). The agile manifesto. Software Development, 9(8), 28-35.
[41] Gobov, D., & Sokolovskiy, N. (2024). An Association Rule Mining for Selection Requirement Elicitation and Analysis Techniques in IT Projects. Software, System, and Service Engineering, 499. DOI: 10.1007/978-3-031-51075-5_4
[42] Aljuhani, A. (2024). Identification of Agile Requirements Change Management Success Factors in Global Software Development Based on the Best-Worst Method. International Journal of Advanced Computer Science and Applications (IJACSA), 15(7). DOI: 10.14569/IJACSA.2024.01507131
[43] Gobov, D., & Sokolovskiy, N. (2023). Association rule mining for requirement elicitation techniques in IT projects. Proceedings of the 18th Conference on Computer Science and Intelligence Systems (FedCSIS), 983–987. DOI: 10.15439/2023F4831
[44] Hakim, H., Sellami, A., & Ben-Abdallah, H. (2024). A Measures-Driven Decision Support System for Managing Requirement Change in Scrum: An Empirical Evaluation. Proceedings of the 19th International Conference on Software Technologies – ICSOFT, 1, 161-168. DOI: 10.5220/0012807700003753
[45] Bantwal, A., & Valilai, O. F. (2026). Integrated engineering change management framework for efficient information flow to product design systems. The International Journal of Advanced Manufacturing Technology. DOI: 10.1007/s00170-025-17175-2
[46] Gonçalves, E., Malcher, P., Antonino, P., & Santos, R. (2025). Investigating requirements change requests. Proceedings of the 21st Brazilian Symposium on Information Systems, 733-742. DOI: 10.5753/sbsi.2025.246624
[47] Gobov, D., & Shevchenko, N. (2024). Definition of the requirements architecture for IT solution as a business analytics product. Innovative Technologies and Scientific Solutions for Industries, 1(27), 26–38. DOI: 10.30837/ITSSI.2024.27.026
[48] Dai, P., Yang, L., Wang, Y., Jin, D., & Gong, Y. (2023). Constructing Traceability Links between Software Requirements and Source Code Based on Neural Networks. Mathematics, 11(2), 315. DOI: 10.3390/math11020315
[49] Rusdianto, D. S., Fabroyir, H., & Yuhana, U. L. (2024). Innovative Approaches to Impact Analysis of Requirement Changes using LLM in Software Projects. IEEE International Symposium on Consumer Technology (ISCT), 604-610. DOI: 10.1109/ISCT62336.2024.10791169
[50] Etezadi, R., Abualhaija, S., Arora, C., & Briand, L. (2025). LLM-driven cost-effective requirements change impact analysis.

arXiv. DOI: 10.48550/arXiv.2511.00262

Authors' Profiles



Dr. Denys Gobov received a master's degree in computer science from the National Technical University of Ukraine "Kyiv Polytechnic Institute" and a Ph.D. in mathematical modeling and computational methods from V.M. Glushkov Institute of Cybernetics of the National Academy of Sciences of Ukraine in 2010. He is currently working as an Associate Professor of the Department of Computer Science and Software Engineering of the National Technical University of Ukraine "Ihor Sikorskyi Kyiv Polytechnic Institute", Kyiv, Ukraine. Denys Gobov has over 20 years of extensive experience in business and system analysis. He has published more than 20 research articles in National and International Journals. Denys Gobov is a vice president on professional development in the Ukraine IIBA chapter, the leading professional association of business analysts. His research area includes business analysis and requirement engineering.



Dr. Oleksandra Zuieva received a Ph.D. in Economics from Simon Kuznets Kharkiv National University of Economics and is currently an Analytics Consultant at CS Ltd., Kharkiv, Ukraine. She has over six years of experience in business and systems analysis and more than two years of teaching experience at the university level. Her research interests include finance, business analysis, and requirements engineering.



Dr. Viktoriia Shashko holds a Ph.D. in economics and currently works at Donbas State Engineering Academy in Kramatorsk, Ukraine. She has more than 25 years of teaching experience in management and business administration, as well as over 10 years in project management. She has published 100 scientific works, including articles in Ukrainian peer-reviewed journals and publications indexed in international scientometric databases, as well as proceedings of international and national conferences. Her research interests include project management, operations management, and education management.

How to cite this paper

Denys Gobov, Oleksandra Zuieva, Viktoriia Shashko, "Context-oriented Framework for Determining Requirements Change Documentation Approaches", International Journal of Information Technology and Computer Science(IJITCS), Vol.18, No.3, pp.96-108, 2026. DOI:10.5815/ijitcs.2026.03.07