Sciences Methods and Technologies International Journal (SciMeTech)

journal homepage: www.scimetech.com

Integrating Low-Code and No-Code Platforms into Complex Enterprise Information Systems: NOVA-LCNC, a Neuro-Symbolic Reference Architecture and Research Agenda

Ikram Souita [1], Younes Zouani [2,3], Abdelmounaim Abdali[1]

[1] Ingénierie et Innovation Technologique des Systèmes et des Procédés (InnovTech), Faculty of Sciences and Techniques, Cadi Ayyad University, Marrakech, Morocco.

[2] Chouaib Doukkali University, Higher School of Technology of Sidi Bennour, Morocco.

[3] Chouaib Doukkali University, LSAM laboratory, Sidi Bennour, Morocco.

How to cite:

Ikram Souita, Younes Zouani, Abdelmounaim Abdali, "Integrating Low-Code and No-Code Platforms into Complex Enterprise Information Systems: NOVA-LCNC, a Neuro-Symbolic Reference Architecture and Research Agenda", Sciences Methods and Technologies International Journal (SciMeTech), (2026) Vol 2, Issue 2, p 84-111

Keywords:
Low-Code and No-Code Platforms
Enterprise Information Systems
NOVA-LCNC
Neuro-Symbolic Architecture
Enterprise Architecture
Governance

Abstract

Low-Code and No-Code (LCNC) platforms are becoming prominent, enabling faster application delivery and strengthening business and IT alignment. When organizations scale these efforts, a significant challenge emerges: enterprise information systems (EIS) are complex environments shaped by heterogeneous architectures, legacy systems, strict governance constraints, and demanding non-functional requirements, including security, performance, reliability, and maintainability. This paper reviews the literature from the perspective of LCNC as a governed abstraction and orchestration layer over core enterprise systems, and uses the resulting synthesis to motivate NOVA-LCNC, a conceptual neuro-symbolic reference architecture for LCNC-EIS integration. The analysis focuses on three themes: (i) LCNC adoption and use, (ii) integration and interoperability in enterprise information systems, (iii) governance, quality, and risk management. The review indicates that the selected corpus provides limited evidence of holistic integration frameworks that relate LCNC adoption to enterprise architecture and established software engineering and governance mechanisms. In response, this paper introduces NOVA-LCNC, a preliminary conceptual architecture that translates business intent into capability-level architecture alternatives. To achieve this, it combines a scoped Contextual Enterprise Twin, candidate synthesis, symbolic constraint verification, counterfactual analysis, Pareto-based explanations, and continuous assurance. We illustrate the architecture through an ERP use case in which users submit purchase requests and monitor budgets through a workflow. At this stage, the work remains conceptual, while implementation and empirical validation are left for future research.

I. Introduction

There is an increasing demand for the rapid delivery of digital solutions, even with constrained development resources and rapidly evolving business requirements.

From a strategic perspective, Low-Code and No-Code (LCNC) platforms can help deliver applications faster, simplify development, and promote a more collaborative relationship between IT teams and business teams.

At the same time, enterprise information systems (EIS) have become increasingly structurally complex.

They are often characterized by heterogeneous technologies, legacy elements, multiple data sources, and tightly controlled environments.

These systems are subject to strict governance constraints and high security, performance and maintainability expectations.

Thus, the introduction of Low-Code and No-Code (LCNC) platforms into the context of an enterprise information system (EIS) is not only a matter of tooling, but also involves integration challenges at the architectural and organizational levels.

The LCNC literature remains uneven. Although benefits, drivers and inhibitors of adopting LCNC and platform limitations have been extensively discussed in previous research, the issue of rapid LCNC adoption without an integration framework in complex enterprise environments is still inadequately explored in a consolidated and operational manner.

To address this, this paper provides a structured and critical examination of the literature on existing practices and frameworks for integrating LCNC platforms into complex enterprise information systems (EIS), and highlights architectural, governance and software engineering perspectives. The review is based on 30 key publications structured around three complementary pillars: (i) adoption and usage, (ii) integration and interoperability, (iii) risks, quality, and governance.

The central argument of this paper is that LCNC platforms should not be understood only as productivity tools, nor merely as alternatives to traditional software development. Rather, they can be viewed as an abstraction and orchestration layer that governs business-side variability while preserving control over core enterprise capabilities. NOVA-LCNC operationalizes this view conceptually by decomposing a business need into capabilities, generating alternative LCNC-EIS allocations, and verifying them against enterprise constraints before human selection.

This paper should therefore be understood as a problem-oriented and conceptual contribution. NOVA-LCNC is proposed as a reference architecture and a conceptual decision support architecture compiler, not as an implemented autonomous system. It defines the problem and the mechanisms to be refined through a prototype, expert evaluation, and enterprise case studies.

Contributions. This work makes five main contributions:

(1) A structured review of the LCNC-EIS literature focused on integration challenges in complex systems,

(2) A thorough analysis of existing constraints that go beyond the typical advantages of LCNC platforms,

(3) The identification, within the reviewed corpus, of a need for more holistic integration frameworks that position LCNC as a governed abstraction and orchestration layer within complex enterprise information systems.

(4) A structured research agenda to guide future research and support sustainable LCNC integration in enterprise contexts.

(5) A preliminary conceptual neuro-symbolic reference architecture, NOVA-LCNC, that links intent interpretation, a scoped enterprise context, capability decomposition, architecture candidate generation, symbolic verification, counterfactual analysis, Pareto-based explanation, architecture compilation, and runtime reassessment.

The conceptual contribution of this paper lies in bringing together capability-level decomposition, scoped enterprise evidence, candidate architecture synthesis, deterministic enterprise constraints, counterfactual scenarios, and an Assurance Envelope within one conceptual LCNC-EIS model, rather than treating adoption, architecture, governance, and lifecycle as separate issues.

The review is guided by these questions: RQ1. What is the current state of the art on the adoption, integration, governance, quality, and lifecycle issues in enterprise contexts? RQ2. What gaps remain when LCNC is examined in the context of complex enterprise information systems? RQ3. Which conceptual mechanisms can support the generation, verification, comparison, and continuous assurance of LCNC-EIS integration architectures?

Paper structure.

- Section 2 introduces the core concepts of EIS complexity, Low-Code versus No-Code, integration and interoperability, and architecture.
- Thematic organization of the review and methodology are described in Section 3.
- Section 4 brings together the work identified across the three thematic dimensions.
- Section 5 discusses the identified literature gaps, develops the NOVA-LCNC reference architecture, and illustrates its application through an ERP-integrated purchase request and budget-control workflow.

- A research agenda is suggested in Section 6.
- Finally, the paper ends with contributions, limitations and directions for future research.

II. Background and key concepts

a. Sources of Complexity in Enterprise Software

Enterprise information systems (EIS) are defined as socio-technical environments that are interconnected and consist of applications, data repositories, stores, services, infrastructure, and governance mechanisms that underpin core organizational processes. EIS complexity goes beyond purely technical aspects and extends to organizational and regulatory dimensions. Typically, it is caused by heterogeneous architectures with various technology stacks, vendors, and integration patterns, legacy dependencies that hinder change, distributed data ownership and multiple sources of truth, and strict operational requirements such as auditability, compliance and service-level requirements.

Security, performance, scalability, reliability, maintainability, and demanding non-functional requirements (NFRs) are critical factors in enterprise systems that are deployed in complex environments. However, while solutions can be rapidly delivered to meet functional requirements, there are significant governance, monitoring, security, and enterprise scalability considerations. As a result, the successful integration of LCNC platforms into enterprise information systems (EIS) must also include proper architectural fit, interoperability, lifecycle management, and governance.

i. What are some of the complexities that make enterprise software so difficult?

A complex enterprise system is not merely a large system, nor simply a system with a large amount of code. It becomes complex when technical, organizational and regulatory components combine in a way that makes change difficult to anticipate. In practice, even the simplest of business requests can travel across various parts of the system and impact interfaces, APIs, databases, permissions, testing, deployment, monitoring, and support. The integration of LCNC should not be regarded as a mere productivity shortcut; rather it should be treated as a carefully managed architectural choice.

Consider an example as simple as moving a database to the cloud; it is not merely about storage. It also influences data consistency, access rules, performance, backups, connected systems, testing environments, rollback strategies, and audit requirements. Similarly, a workflow can impact forms, permissions, APIs, dashboards, notifications, and logs. These examples illustrate an important consideration: enterprise complexity is not necessarily located within a single component of the system; it lies in how components interact.

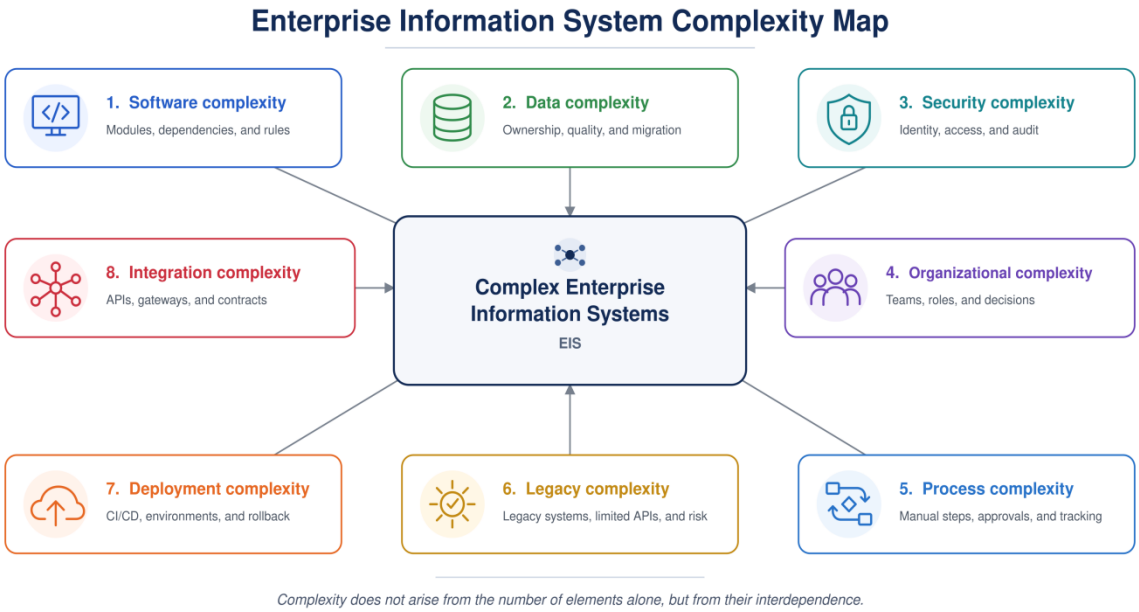

Figure 1: Enterprise Information System (EIS) Complexity Dimensions and Interdependencies

Complexity dimension	Enterprise manifestation	Why it becomes problematic	Possible LCNC contribution	Condition/Limit
Software complexity	Modules, rules, dependencies, interfaces.	A small change may touch front-end, back-end, tests and deployment.	Externalize simple forms, dashboards, approvals, and configurable workflows.	Critical algorithms and transactions remain in engineered services.
Data complexity	Distributed data, master data, migrations, sensitivity.	Duplication and inconsistent sources of truth appear easily.	Provide governed data views and validation screens via APIs.	LCNC must not become the owner of critical data.
Deployment complexity	Systems rely on multiple environments and pipelines.	Small visible changes can require heavy release processes.	Allow faster updates for low-risk configurations.	Core deployments still require strong DevOps practices.
Integration complexity	Systems communicate through APIs, events, and connectors.	Business processes depend on many systems working together	Simplify usage by wrapping APIs into reusable actions	APIs must remain secured, versioned, and governed.
Security complexity	Access control, roles, and compliance are everywhere.	Each new feature increases risk and complexity	Centralize permissions and track actions	Deep security must stay under enterprise control
Legacy complexity	Old systems are rigid and hard to change.	Updates are slow, risky, and expensive.	Add modern interfaces and workflows around legacy.	LCNC cannot remove technical debt.

Table 1: Complexity dimensions in enterprise systems and the role of LCNC with associated limits

b. Low-Code vs. No-Code and Citizen Development

Low-Code and No-Code platforms can be viewed as two points along a continuum of abstraction. Low-Code platforms combine visual modeling and ready-to-use components with extensibility through custom code or sophisticated configuration capabilities. No-Code platforms, on the other hand, seek to enable application development without coding, via composition, templates, and limited configuration mechanisms.

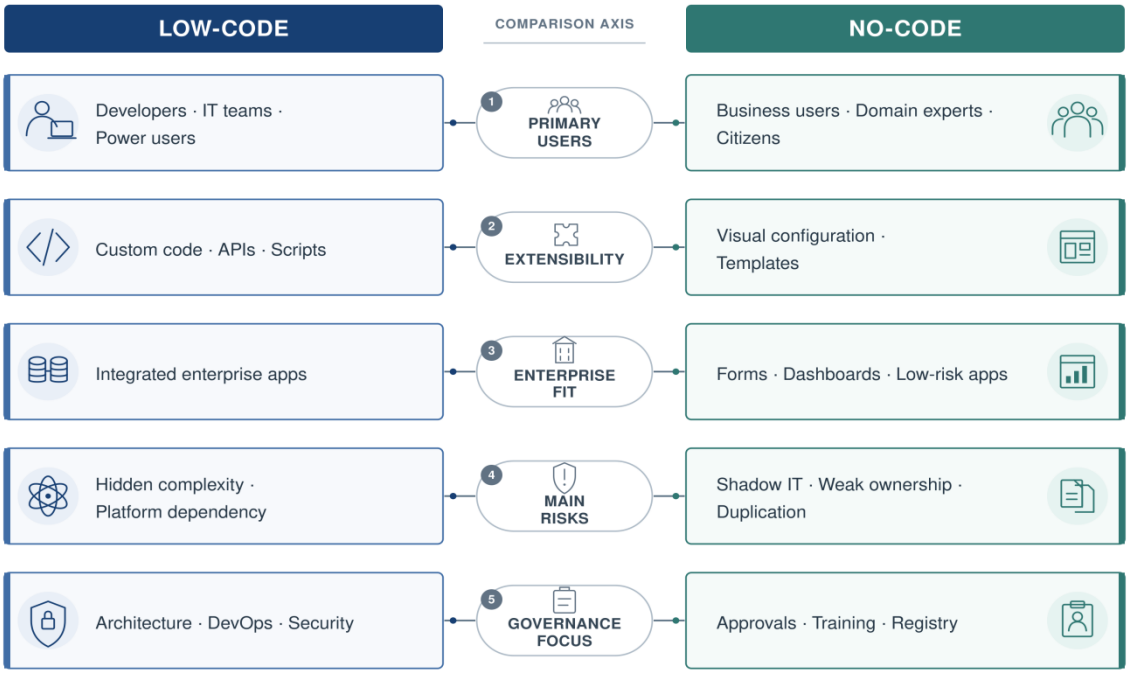

Figure 2: Low-Code versus No-Code differentiation in enterprise contexts

Citizen development refers to the practice of building applications by individuals who are not traditional software developers, including business-unit staff, domain experts, and operations
staff who generally work outside of formal IT delivery processes. Citizen development can help speed up application delivery and ease development bottlenecks in enterprise environments; however, it also presents significant governance challenges such as quality variability, shadow IT, inconsistent data models, and security risks if it is not aligned with enterprise architectures and software engineering.

c. Integration and interoperability in enterprise contexts

In enterprise applications, integration refers to the ability to connect systems, enable data sharing, and enable process synchronization across technical and organizational constraints. Most integration is achieved through technical tools like APIs, middleware event buses, data pipelines, identity and access management (IAM), and monitoring and logging infrastructures. What is therefore fundamental to integration is runtime execution and the reliability of the processes in operation.

Interoperability, on the other hand, is the ability of various systems to interact in a meaningful way, and not only to exchange data, but also to share meaning, model compatibility, and portability between tools or environments. Interoperability may be addressed at multiple levels, including data-level interoperability (schemas, formats, data quality constraints, master data alignment), process-level interoperability (workflows, orchestration mechanisms, business rules), interface-level interoperability (user interface components and interaction logic), and lifecycle-level interoperability (deployment, versioning, testing, and alignment with DevOps practices). In the context of LCNC platforms, integration and interoperability issues are especially salient, given that enterprise deployments frequently depend on interactions with legacy systems, shared databases, external SaaS services, and tightly regulated business processes. Limited control over data models, security configurations, deployment pipelines, or data export and import mechanisms can expose organizations to risks including vendor lock-in, incomplete migration paths, fragmented integration patterns, and reduced observability in production environments. From an integration perspective, LCNC analysis concerns runtime connectivity and operational reliability, while from an interoperability perspective, it addresses portability, semantic consistency, and sustainability.

d. Architectural considerations for LCNC-EIS integration

When integrating LCNC platforms into complex EIS, the architectural choices are directly connected to the impact on scalability, security, maintainability, and governance. The following architectural factors should be considered when integrating LCNC into EIS.

e. Architectural alignment and boundary definition

The placement of LCNC applications in the enterprise information system should be explicitly specified by organizations. This involves determining whether LCNC applications are used as peripheral departmental solutions, integrated into core business processes or used for customer-facing or mission-critical applications. Clear boundaries lessen the risk of uncontrolled proliferation of applications and ensure consistency of integration and governance.

f. Data architecture and ownership

Rapid uptake of LCNC platforms often allows for quicker application development than enterprise data governance practices can support.

To reach architectural alignment, it is essential to define sources of truth, data ownership, acceptable replication patterns, data access control and constraints on local data storage on the LCNC platform. If these boundaries are not defined, then data may be fragmented and may create
inconsistent and conflicting data models within the enterprise.

i. Security and identity integration

Complex enterprise information systems (EIS) usually employ standardized identity and access management (IAM) mechanisms including single sign-on (SSO), multi-factor authentication (MFA), and role or attribute-based access control (RBAC/ABAC), as well as audit trails and compliance controls.

Therefore, it is important that the integration of LCNC platforms ensures that all access is consistently authenticated and authorized, with end-to-end data access traceability and alignment between platform security features and enterprise-wide security policies.

ii. Lifecycle, DevOps, and change management

For enterprise solutions, version control, comprehensive testing, automated deployments, release control, and rollback are required. When it comes to integrating with CI/CD, environment separation, configuration management and automated testing, LCNC platforms vary widely.
Consequently, architectural considerations need to include the promotion of LCNC artifacts across environments, and the governance of changes for production stability.

iii. Observability, reliability, and operational ownership

Sustained operational success in a complex EIS depends largely on thorough monitoring and logging, clear incident management processes, performance monitoring, and clearly defined ownership. For LCNC solutions, the key requirement is the ability to integrate with an enterprise observability stack, or provide equivalent capabilities, and organizations must clearly assign responsibility for incident response and the long-term maintenance of citizen-developed applications.

iv. LCNC as a governed abstraction and orchestration layer

This subsection introduces a central argument that complements and strengthens this article:
LCNC should not be understood as a substitute for software engineering. It is more effective in complex enterprise systems where it is used as a governed mediation layer between fast-evolving business needs and stable enterprise capabilities.

In this scenario, LCNC can help with form configuration, workflows, approvals, dashboards, and service composition, while the enterprise continues to retain control over critical elements such as master data, identity, business logic, APIs, security, and deployment practices.

This view highlights the core argument of this research. LCNC does not remove complexity; it shifts some business-side variability to the forefront as a configurable and manageable layer. When this layer is not well controlled and managed, however, it can add new hidden complexities, such as shadow IT, duplicated data, fragile integrations, undocumented workflows, and unclear ownership.

Specifically, the LCNC layer proposed in this article is defined by four boundaries. First, LCNC can support business-side variability, for example, forms, workflows, approvals, dashboards, and basic service composition. Second, critical enterprise capabilities, such as master data, identity, security enforcement, core business logic, and

performance-sensitive processing, should remain outside the LCNC layer. Third, all interactions between LCNC and core systems should be mediated through governed interfaces, such as APIs, connectors, IAM mechanisms, logging, and monitoring. Fourth, each LCNC initiative should be associated with ownership, approval, documentation, lifecycle, and audit controls.

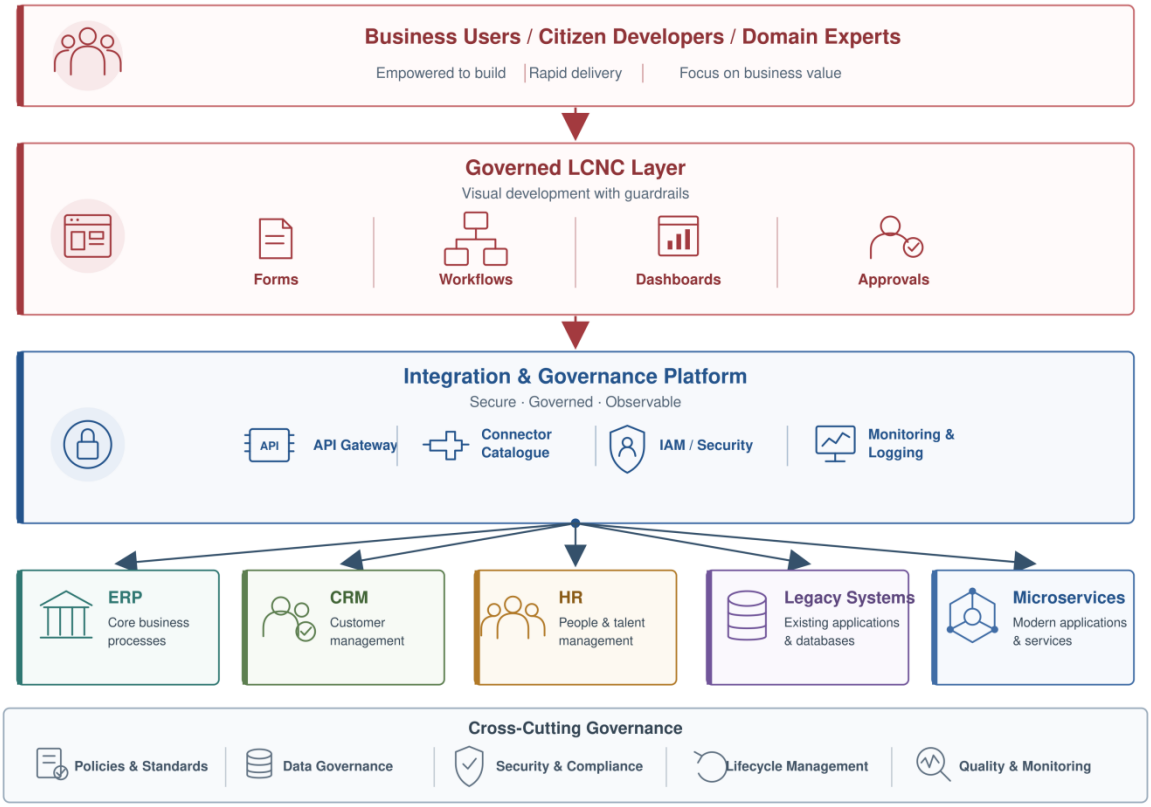

Figure 3: Conceptual architecture of a governed LCNC layer within complex enterprise information systems (EIS)

The proposed role of LCNC as a governed abstraction and orchestration layer is illustrated in Figure 3. Moving core enterprise logic to LCNC platforms is not the main objective of LCNC.
Instead, the goal is to use LCNC as a platform for controlled business variability. Enterprise architecture and software engineering controls remain in place for forms, workflows, dashboards, approvals, and simple service composition, while critical business logic, performance-sensitive processing, master data, identity, and security enforcement remain under stronger control.

g. Section summary

This section has specified the elements needed to analyze LCNC integration within complex enterprise information systems (EIS). It emphasized that EIS complexity has a socio-technical dimension, which includes organizational, technical, and regulatory constraints. The difference between low-code and no-code platforms in terms of extensibility and control has direct consequences on the feasibility of integration. Integration and interoperability are different but complementary aspects. Coherent architectural alignment of the data, security, lifecycle, and operational aspects is necessary for sustainable adoption of LCNC. All these factors drive the structured review methodology and thematic synthesis discussed in Section 3.

III. Research methodology

a. Review approach

This paper is based on a literature review. The goal was to examine relevant studies, compare them, and understand their contributions to low-code and no-code research. This review does not attempt to address all the individual papers on the subject. A comprehensive review of all publications is beyond the scope of this paper. Instead, it focuses on the studies that are most relevant for this research.

The literature search was conducted using several academic databases and digital libraries, including IEEE Xplore, ACM Digital Library, ScienceDirect, SpringerLink, Google Scholar, arXiv, MDPI, AIS eLibrary, and SSRN. The aim was to identify relevant studies dealing with Low-Code and No-Code solutions in relation to enterprise

information systems, integration, interoperability, governance, citizen development, software quality, and software lifecycle management.

The review covered the period from 2020 to 2025, allowing it to capture recent advances and relevant publications within the defined timeframe. The search strategy relied on various combinations of keywords, including "low-code", "no-code", "LCNC", "enterprise information systems", "enterprise architecture", "integration", "interoperability", "governance", "citizen development", "software quality", "DevOps", "platform selection", and "AI-assisted no-code".

Studies were selected according to well-defined inclusion and exclusion criteria. Publications were considered when they addressed the use of Low-Code/No-Code platforms in one of the following areas: adoption, enterprise use, integration, interoperability, governance, software quality, citizen development, or lifecycle management. However, papers that were primarily commercial, tutorial-oriented, not related to enterprise contexts, based on isolated prototypes, or offered limited analytical or research value were excluded.

After the removal of duplicates and the screening of titles, abstracts, and full texts, 30 publications remained in the final corpus. The results of these studies were subsequently grouped into three thematic streams: adoption and usage; integration and interoperability; and governance, quality, and risk management.

A total of 68 records were identified, including 58 records from databases, publisher platforms, and repositories, and 10 additional records through reference screening. After duplicate removal, 61 records were screened by title and abstract; 20 records were excluded, 41 full-text articles were assessed for eligibility, and 30 publications were retained in the final corpus.

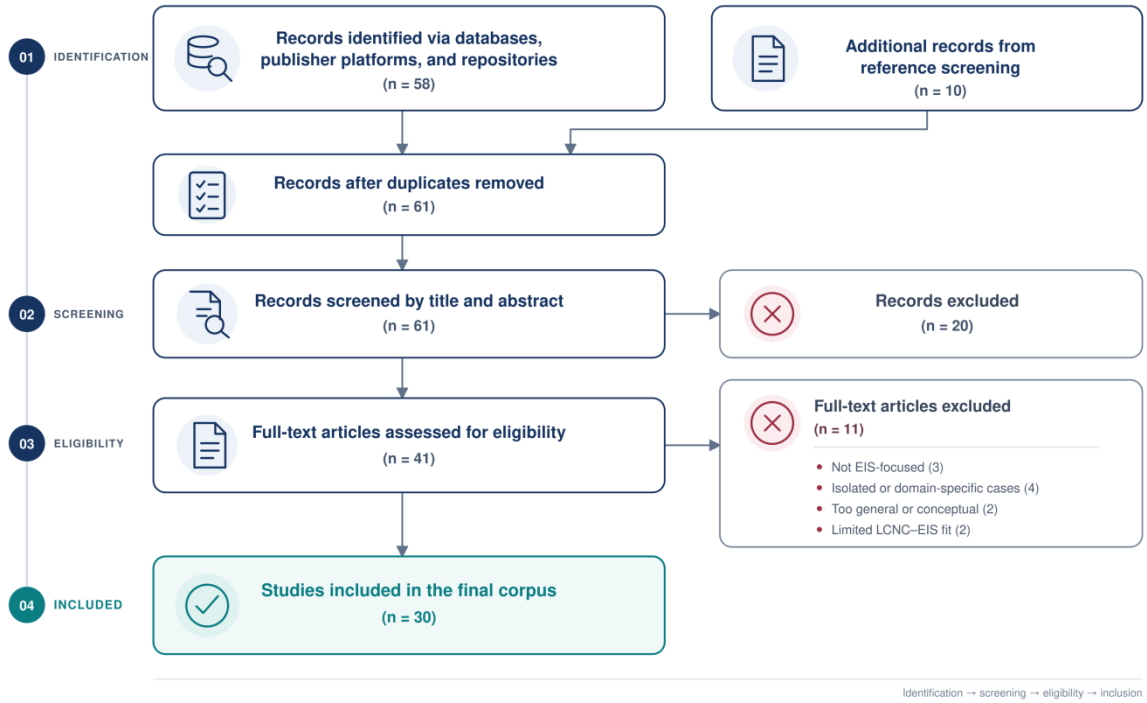

Figure 4: PRISMA-inspired flow diagram of the literature selection process

This approach is appropriate for the paper because of the following. First, the LCNC field is still fragmented. Researchers may not employ the same terminology, nor always the same types of study. Some papers focus more on adoption and readiness. Others focus on usability, integration, portability, governance, or the development of no-code and AI-assisted no-code. Second, this paper does not merely ask whether LCNC platforms are useful; it addresses a more challenging question: How can these platforms be adopted in a stable manner within enterprise environments that are complex? They tend to be challenging environments that have a mix of legacy systems, varying architectures, regulatory requirements, and technical requirements such as security, performance, scalability, and maintainability.

For this reason, the review will not consider LCNC solely as a means of saving time or making development easier. It adopts a business perspective on LCNC. In other words, the review aims to provide an integrated discussion about the adoption, interoperability, governance, software quality and lifecycle management aspects.

i. Conceptual analytical lens for enterprise complexity

This study introduces a conceptual analytical lens to connect the literature review with the enterprise complexity perspective adopted in this paper. This lens is useful when considering the scope of existing enterprise architecture: whether the context involves a monolithic system, legacy workflow systems, or involves low-risk scenarios for consistent changes in the business process where a full new microservice is not warranted. This is not an alternative to literature review; it complements the literature review by demonstrating the applicability of the concepts discussed to real-life enterprises.

b. Corpus construction and selection logic

The review is based on the analysis of 30 publications. For the analytical purposes, they were subdivided into three major groups.

The first group relates to adoption, usage, business-IT collaboration, and citizen development [1]-[12]. It includes studies on low-code adoption, real-life evidence, ease of use, no-code usage, as well as subjects such as business-IT transformation, no-code product development, democratization, citizen development strategy, among other topics.

The second group relates to integration, architecture, interoperability, portability, platform selection, model-driven approaches, DevOps views, and recent studies on AI-assisted and LLM-based no-code systems.

The third relates to aspects of governance, quality, risk, roles, and enterprise constraints [25]-[30]. It contains research on citizen development governance benefits and constraints in various sectors, hidden platform-related issues, delivery quality, development efficiency, role changes, and software quality based on ISO/IEC 25010:2023.

Four main criteria were used to select the publications. First, they needed to be closely associated with low-code, no-code, or organizational or enterprise use. Second, they had to cover at least one important topic in this paper, such as adoption, integration, interoperability, governance, quality, or citizen development. Third, they needed to go beyond prototype cases or purely technical explanations of LCNC. Fourth, they had to be of sufficient academic quality and peer-reviewed or have been presented in a conference (with details) or be available on arXiv (with clear research value) or be published locally (full-text) and relevant for the no-code discussion.

The resulting corpus includes both low-code studies and an identifiable no-code group, providing a more balanced perspective on LCNC. This balance is important because no-code presents distinct challenges related to citizen development, business autonomy, platform dependency, and the recent growth of natural-language-based application creation.

Table 2 provides a structured summary of the selected publications according to their reference numbers, year, publication type, thematic focus, and main contribution to the review.

Ref	Year	Type	Theme	Main contribution
[1]	2025	Systematic literature review	Adoption	Synthesizes existing work on LCNC adoption by bringing together the main benefits, challenges, theoretical lenses, and open research directions.
[2]	2022	Conference paper	Adoption	Highlights the factors that can encourage or limit the adoption of low-code development platforms in organizational settings.
[3]	2023	Empirical study	Adoption	Explores how practitioners perceive the adoption of low-code platforms and which factors they consider most relevant in practice.
[4]	2023	Journal article	Adoption/ Digitalization	Discusses how low-code and no-code platforms can support the digitalization of management processes, particularly in SME contexts.
[5]	2023	Empirical software engineering study	Adoption/ Barriers	Studies developer discussions to identify recurring concerns around platform adoption, integration, customization, data management, and long-term maintenance.
[6]	2023	Systematic literature review	Usability	Reviews usability-related challenges in low-code platforms and shows how user experience remains an important issue for broader adoption.
[7]	2024	Empirical study	No-Code / Developer experience	Compares the use of no-code platforms with developers' experience, focusing on perceived productivity, efficiency, and practical limitations.

[8]	2021	Journal article	No-code development	Provides an example of no-code application development and illustrates how platform-based tools can simplify application creation.
[9]	2023	Journal article	Business-IT relationship	Examines how no-code platforms can change collaboration patterns between business users and IT specialists.
[10]	2023	arXiv paper	No-code product development	Discusses the role of no-code tools in digital product development and their potential impact on how products are designed and delivered.
[11]	2025	Journal article	No-code development	Introduces a drag-and-drop no-code platform intended to make application development more accessible to non-specialist users.
[12]	2024	Journal article	Citizen development	Proposes a citizen development strategy supported by LCNC platforms, with attention to organizational use and implementation conditions.
[13]	2021	Conference paper	Knowledge integration	Shows how low-code platforms may help connect business knowledge and technical implementation within collaborative development processes.
[14]	2020	Conference paper	Platform comparison	Identifies comparison dimensions that can be used to better understand differences between low-code development platforms.
[15]	2024	arXiv paper	Interoperability	Examines the technical difficulties related to interoperability and portability when applications move across low-code platforms.
[16]	2025	arXiv paper	Platform selection	Suggests a structured way to evaluate low-code platforms and support more informed platform selection decisions.
[17]	2022	Multivocal systematic review	Model-driven engineering	Reviews how low-code development relates to modeling practices and discusses connections with model-driven engineering.
[18]	2022	Journal article	Model-driven engineering	Analyzes the relationship between low-code development and model-driven engineering, especially in terms of abstraction and model use.
[19]	2022	Conference paper	DevOps/Lifecycle	Investigates how DevOps practitioners understand the rise of low-code development and its implications for software delivery practices.
[20]	2025	Journal article	AI-assisted no-code	Presents an AI-supported no-code environment that allows users to build web applications through language-model assistance.
[21]	2025	Journal article	API generation	Proposes a no-code approach for generating REST APIs from natural language descriptions, reducing the need for manual backend coding.
[22]	2025	arXiv paper	LLM-based no-code	Explores how large language models and Function-as-a-Service can be combined to support no-code application development.
[23]	2025	arXiv paper	AI-assisted workflow building	Presents a workflow-building approach where natural language and multi-agent collaboration help non-experts design automated processes.
[24]	2025	arXiv review	Zero-code/LLM tools	Reviews recent zero-code tools based on large language models and discusses their use for application development by non-programmers.
[25]	2024	Journal article	Governance	Discusses governance mechanisms needed to manage citizen development and address challenges associated with low-code platform use.

[26]	2023	Journal article	Sector-specific digitalization	Examines how low-code development can support digitalization in the construction sector, while also pointing out its practical limitations.
[27]	2021	Practitioner article	Risk/Hidden complexity	Draws attention to the hidden technical and organizational issues that may remain behind apparently simple low-code and no-code solutions.
[28]	2024	Research paper	Quality/ Efficiency	Studies how LCNC programming can influence development efficiency and delivery quality in software production contexts.
[29]	2024	Conference paper	Roles/ Organization	Identifies new roles and organizational responsibilities that emerge when low-code platforms become part of software development practices.
[30]	2025	Journal article	Software quality	Assesses LCNC platforms using ISO/IEC 25010:2023 quality characteristics to better understand their strengths and weaknesses from a software-quality perspective.

Table 2: Summary of the Selected Publications

c. Analytical procedure

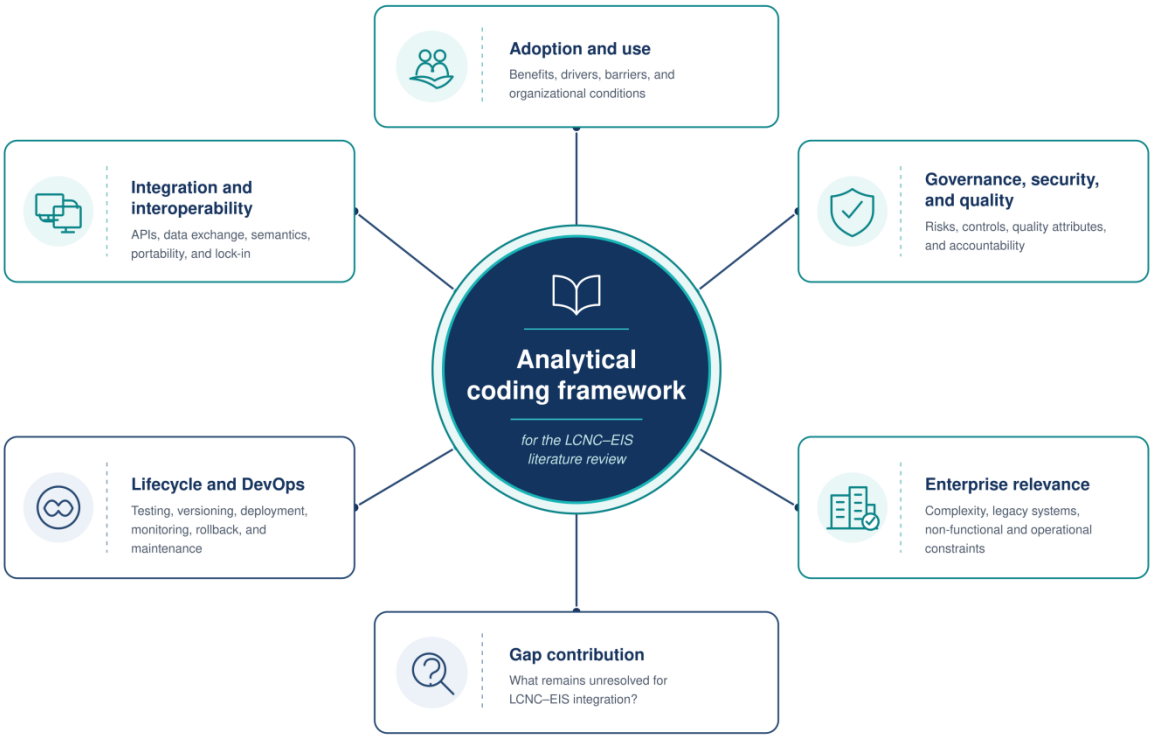

Figure 5: Analytical coding scheme used for thematic synthesis

All publications were read in the same manner. In each paper, the analysis examined four aspects: the research goal, the method used, the main findings, and what each paper indicates about LCNC integration in complex enterprise environments.

The analysis was then conducted in two steps: First, the papers were analyzed within each group for the presence of common ideas, tensions, and gaps. Second, a cross-theme analysis was undertaken to examine how adoption, interoperability, governance, and quality interact when LCNC is seen not just as a tool, but as part of a broader enterprise change.

This analysis went beyond a simple summary of each paper. The intent was to uncover deeper themes in the literature. For instance, in the realm of studies on adoption, the concepts of speed, accessibility, and empowerment

appear frequently. Alongside these strengths, however, lie risks in more challenging enterprise contexts, as revealed in studies of integration and governance. In contrast, a similar pattern can be observed in AI-powered no-code research. While these tools help make development more accessible, they also introduce new questions of validation, portability, and governance of generated artifacts.

d. Methodological limitations

This study has certain limitations. First, it is not exhaustive. It is not a comprehensive SLR or meta-analysis. Second, the LCNC field is still fragmented. Low-code, no-code, citizen development, model-driven engineering, interoperability, and platform governance are still discussed as distinct concepts. Third, there is no uniformity in the degree of maturity of the literature.

There are existing systematic reviews and empirical evidence in some low-code areas, with numerous studies of no-code and AI-assisted no-code being more exploratory.

Despite these limitations, the review supports a central argument of the paper: the field is growing, but the reviewed corpus provides limited evidence of integrated and practical frameworks that connect adoption, enterprise architecture, interoperability, governance, lifecycle practices, and software quality in a coherent manner.

IV. Literature analysis

a. Adoption, usage and business-IT transformation

As presented in the literature, the adoption of LCNC cannot be limited to a mere technological preference. Previous research and reviews have shown that organizational, platform, perception, and context-specific factors have an impact on adoption [1]-[4].

Ajimati et al. [1] provide an important synthesis that demonstrates that key themes that emerge in the literature on LCNC research include benefits, challenges, theoretical lenses, and future research directions. Nevertheless, their review also shows that the field is not fully coherent, and that adoption is still frequently disconnected from enterprise architecture and governance issues.

This fragmentation is more apparent in the work of Kass et al. [2], [3] who show that there are several drivers and barriers to the adoption of LCNC, and that the importance of these drivers and barriers varies by context. This is significant because LCNC adoption cannot be assessed abstractly. While some platforms may seem like suitable choices in some organizations, they might not work well in others based on governance maturity, security sensitivity, and organizational readiness. This interpretation is reinforced by Domanski et al. [4], who demonstrate that the adoption of platforms is still very much context-dependent, even in SME contexts.

Practitioner and usability perspectives further enrich this picture. As revealed in developer discussions in Al Alamin et al. [5] adoption and day-to-day use quickly reveal problems involving APIs, data handling, customization, and maintenance. Additionally, Pinho et al. [6] state that the conceptual basis of low-code remains partially unstable, especially with regard to definitions and the treatment of usability.

The significance of these findings lies in the complexity of enterprise environments where the seemingly simple nature of LCNC can reveal more complex issues that become visible in practical use.

The no-code literature adds to this picture of adoption but complicates it as well. Guthardt et al. [7] demonstrate that no-code adoption is not just about the technical ability of non-developers to use it, but also whether the organization can support training, flexibility, and requirements fit.

Studies devoted to the no-code concept demonstrate that no-code is changing the business-IT relationship as described in [8]-[12]. The organizational aspect of this shift is emphasized by works like studies on no-code development platforms and LCNC-enabled citizen development strategies. No-code is therefore not just about building without code; it is also about redefining who builds, how business actors interact with technical functions, and how responsibility is redistributed.

Concurrently, the literature about no-code product development explicitly highlights the main advantages of no-code, which are related to speed, autonomy, and low entry barriers [10], [11]. These are real benefits, particularly during the initial stages of product creation or departmental solution building. But these benefits are not uniform from an enterprise point of view. The same democratizing power that makes no-code appealing can also create disorganized practices, dependency on the platform, and lack of enterprise standards when governance and architecture are not robust enough.

The adoption literature indicates that LCNC should rather be viewed as a socio-technical change than simply a tool selection decision. Both low-code and no-code are driven by the same intent: speed and accessibility, but have different focuses. While extensibility and enterprise integration are usually more closely associated with low-code, business empowerment, citizen development, and lower technical barriers to entry are more closely associated with no-code [1]-[4], [7]-[12]. This is important because it shapes the governance and integration of each stream within a complex enterprise information system.

b. Integration, architecture and interoperability

The second line of literature clearly indicates that the real enterprise challenge begins when LCNC applications need to coexist with existing systems, data architectures and operational requirements. Iho et al. [13] offer an important conceptual contribution that low-code platforms can support knowledge integration between business and IT beyond delivery acceleration. This insight is useful in enterprise situations where LCNC is used not just as a productivity tool but as a coordination and alignment mechanism.

Comparative and evaluative studies emphasize this aspect of the enterprise dimension. Sahay et al. [14] list some important dimensions of comparison of platforms, including interoperability, security, collaboration, reusability, and scalability. Lamanna [16] suggests a more operational perspective by introducing an evaluation framework that explicitly considers integration, interoperability, governance, and security. Together, these studies support a central idea of this paper: LCNC platform selection is not a secondary procurement choice, but an enterprise architecture decision.

One of the most important and problematic gaps in practice and the literature is highlighted by interoperability work. Alfonso et al. [15] demonstrate that for data export and import, portability is still relatively strong, but semantic and behavioral portability is often weaker. This introduces risks of vendor lock-in, and migration is more difficult than the discourse of platform flexibility implies. For complex enterprise systems, portability is not only a technical consideration, but also a strategic consideration related to organizational autonomy, maintainability, and the ability to adapt systems over time.

Further understanding of the problem can be gained from the model-driven engineering literature. Bucaioni et al. [17] demonstrate that low-code is related to model-driven thinking, although the field is not homogeneous, and sometimes overly optimistic. Di Ruscio et al. [18] argue that low-code and model-driven engineering are not synonymous. LCNC platforms are not just modeling tools, but they also include lifecycle logic, deployment logic, and ecosystem logic. In an enterprise context, integration is not just about runtime connectivity, but also deployment, testing, versioning, and maintainability.

Rafi et al. [19] build on this view by demonstrating that low-code is viewed as an aspect of delivery industrialization and DevOps. This is particularly important for enterprise information systems that are complex enough to require applications to be tested, promoted and observed across environments and then maintained in production if they are to be considered sustainable. This means that integration cannot be separated from its operationalization.

The no-code and AI-assisted no-code literature takes this integration discussion in a new direction. The emergence of NoCodeGPT [20], SmartAPIForge [21], LLM4FaaS [22], AIAP [23], and the review of zero-code LLM-based tools [24] indicates that no-code is transitioning towards natural-language-driven generation, automated API creation, multi-agent orchestration, and function-based assembly. While these innovations increase the power and accessibility of no-code, they also present additional questions for enterprise integration. Although generated applications may be easier to produce, they are more difficult to validate, port, secure, or observe. The core of the integration issue remains; however, it becomes more dynamic and, in some instances, less transparent.

Taken together, this second body of literature reveals that integration of LCNC in enterprise environments needs to be considered at several levels: technical connectivity, semantic consistency, platform portability, lifecycle compatibility, and delivery governance [13]-[24]. Many of the above questions are already explicitly covered by low-code literature, but the no-code literature increasingly raises the above questions as well, particularly when the no-code systems become more complex and more tightly coupled with AI-assisted generation.

c. Governance, quality, roles and risk

The third literature stream makes clear that LCNC democratization is a governance-dependent process. One of the most solid foundations here is provided by Viljoen et al. [25], who observe how citizen development can present several risks for enterprises, including shadow IT, quality degradation and technical debt, if organizations fail to set appropriate control mechanisms.

This contribution is important, as the governance of LCNC is not just an extension of traditional software governance. It is based on platform affordances, the relationship between experts and non-experts, and the design of supervision mechanisms.

This is supported by sector-specific and professional views. Martinez and Pfister [26] found that low-code could enable process digitalization, but it requires a strategic rather than ad hoc approach.

From a practitioner perspective Hurlburt [27] agrees that although at the surface level, LCNC simplifies development, technical understanding is not eliminated. These studies are helpful because they illustrate a consistent theme: LCNC does not erase complexity, it shifts it.

The argument is further enriched by quality studies. Cui [28] stresses the balance between speed and long-term requirements like scalability, customization, and maintainability. Naqvi et al. [30] offer a more formal quality lens by linking LCNC assessment to ISO/IEC 25010:2023.

This is particularly useful in enterprise contexts where the discussion shifts from general statements about efficiency to quality dimensions like reliability, compatibility, maintainability, and security.

Role transformation is a key governance issue. Mottu and Sunyé [29] demonstrate that the responsibilities of organizations are shifting because of LCNC, with roles like platform engineer and citizen developer becoming more prominent. When combined with the no-code literature [8]-[12] and AI-assisted no-code systems [20]-[24], which expand the list of potential builders even more, this observation becomes even more significant. This is not just about increased output. It also creates a greater need for clarity of role, accountability, review, and escalation processes.

One of the biggest lessons learned from this stream is that governance, quality and role design cannot be addressed after implementation. They should be built into the LCNC adoption logic from the start. This holds true for low-code, where enterprise extensibility and interoperability come into play, and even more so for no-code, where enterprise accessibility and democratization can outpace control. In both instances, the boundary definition, validation, monitoring, and embedding of platforms into organizational structures are crucial for enterprise sustainability [25]-[30].

d. Cross-theme synthesis

A clear pattern is evident in the thirty publications. The benefits of LCNC (faster delivery, closer business-IT collaboration, lower barriers to creation, and perceived flexibility) are explained in the adoption studies [1]-[12]. Integration studies indicate why these same qualities become problematic in complex enterprise information systems: weak portability, partial interoperability, lifecycle misalignment, and platform dependency [13]-[24]. Over time, these tensions pose risks highlighted by governance and quality studies: shadow IT, technical debt, unclear roles, weak control, and uneven software quality [25]-[30].

Three implications arise from this synthesis. First, low-code and no-code occupy a shared enterprise problem space, but they do not behave identically. In the reviewed corpus, low-code is more prominent in enterprise integration, while no-code appears more frequently in business empowerment, citizen development, and AI-based development discussions. Second, the literature remains fragmented across these streams. Third, the reviewed corpus provides limited evidence of an LCNC-EIS integration perspective that connects adoption, architecture, interoperability, governance, quality, and organizational transition in a single model.

i. LCNC shifts complexity; it does not erase it

One of the major lessons learned from the reviewed literature is that LCNC can decrease the apparent complexity for business users but can also increase enterprise complexity if not managed properly. NOVA-LCNC therefore considers each capability individually, rather than treating the entire application as either LCNC-based or traditionally coded. Business-facing variability can be managed through LCNC, while distributed transactions, master-data ownership, security enforcement, performance-critical logic, and core algorithms remain handled through conventional software engineering.

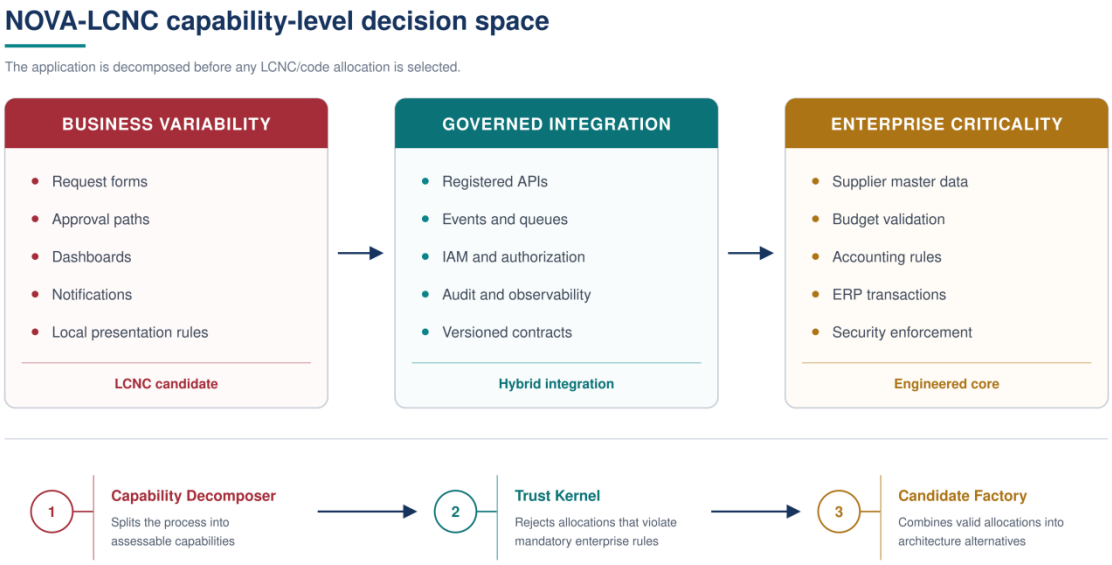

Figure 6: Capability-level decision space used by NOVA-LCNC in complex enterprise information systems

As illustrated in Figure 6, NOVA-LCNC defines the decision space at the capability level. Approval workflows, dashboards, and notifications may be suitable LCNC candidates. Governed APIs and events, together with IAM, auditing, and observability, form the integration space, while critical transactions, master data, security enforcement, and performance-sensitive logic remain within the engineered enterprise core.
These initial allocations are not treated as final decisions.

capability-level decision question	LCNC-oriented answer	Engineered/core allocation
Is the capability visible, configurable, and frequently changing?	LCNC is a candidate when changes remain governed.	Code preferred when deep customization or high performance is required.
Does the capability execute a critical transaction or modify master data?	LCNC may only invoke a controlled operation.	The transaction and data ownership remain in ERP or authorized services.
Can the capability use a registered API or event contract?	LCNC may consume the governed interface.	A central integration service is required when no governed interface exists.
Does the capability enforce sensitive security or compliance rules?	LCNC inherits IAM, logging, and policy controls.	Security-sensitive enforcement remains centralized and independently testable.

Table 3: Decision criteria for LCNC versus traditional code.

e. Proposed preliminary LCNC-EIS integration framework

This section connects the gaps found in the previous analysis with the proposal of a preliminary LCNC-EIS integration framework. The framework is not meant to replace existing enterprise architecture, DevOps, or security engineering practices. Rather, it explains how LCNC platforms can be deployed, managed, and used within a complex enterprise information system. The main idea is that LCNC should support business-side variability through governed forms, workflows, dashboards, approvals, and simple service composition, while architecture and engineering should keep tighter control over critical enterprise capabilities.

f. A rich but fragmented field

The results of this review indicate that LCNC research is not only extensive but is also still developing in an uncoordinated manner as a single field. Many studies discuss adoption [1]-[4], [12] usability and practitioner views [5], [6], interoperability [14],[18], no-code democratization and product development [8]-[11], and governance and quality issues [25]-[30]. However, these studies usually examine these topics separately rather than integrating them into a single perspective. As a result, the field appears to be active and promising yet fragmented. It can easily be argued that this issue is evident when low-code and no-code are grouped together, despite their actual differences, which are not always studied in detail, particularly in complex enterprise environments.

g. A persistent low-code bias in enterprise research

A key finding of this review is that enterprise research is much more oriented toward low-code rather than no-code. More detailed studies are available on low-code for topics like platform comparison, interoperability, model-driven engineering, DevOps alignment, governance and quality [14]-[19],[25]-[30]. By contrast, accessibility, business autonomy, startup product development, and more recently, AI-assisted generation are more commonly associated with no-code [8]-[12],[20]-[24]. This distinction is crucial because it indicates a lack of research into no-code in enterprise integration. It is less developed in the literature compared with low-code research.

h. From a missing holistic framework to NOVA-LCNC

NOVA-LCNC is structured around three questions. First, which capabilities make up the business need, and how should each capability be allocated across LCNC, governed integration, and the engineered enterprise core? Second, which enterprise requirements must the candidate architectures satisfy? Third, which operational and organizational conditions are required to keep the selected architecture continuously assured?

Together, these questions shift the proposal from a static structure to a conceptual process of architecture synthesis, verification, explanation, and continuous reassessment.

In addition to the limited availability of comprehensive LCNC-EIS integration frameworks in the reviewed corpus, the review did not identify a conceptual mechanism that fully connects enterprise context, capability decomposition, alternative generation, constraint verification, future scenarios, governance outputs, and runtime reassessment. The literature provides useful individual elements, but few studies bring them together into a coherent decision-making process.

NOVA-LCNC addresses this gap through a neuro-symbolic architecture. AI-assisted functions interpret business intent, identify the relevant enterprise context, generate candidate architectures, and explain their trade-offs. A symbolic Trust Kernel enforces explicit rules related to architecture, data, security, and lifecycle, while human decision-makers provide missing information and select among acceptable alternatives. Within this approach, LCNC remains a governed abstraction and orchestration layer and does not own critical enterprise capabilities.

i. Proposed preliminary NOVA-LCNC reference architecture

NOVA-LCNC is presented as a preliminary conceptual reference architecture. It is neither an automated compiler nor a substitute for established enterprise architecture methods. In this paper, the term "architecture compiler" is used as shorthand for a transformation chain that translates explicit business intent into verified architecture alternatives, together with governance and assurance artifacts. Implementation and empirical validation remain subjects for future work.

The nine conceptual modules of NOVA-LCNC and the main information flows between them are shown in Figure 7. The architecture covers intent formalization, the definition of a scoped enterprise context, capability decomposition, candidate solution generation, deterministic verification, counterfactual analysis, Pareto-based decision support, architecture compilation, and runtime reassessment.

Within this preliminary conceptual architecture, the term "neuro-symbolic" denotes a separation of responsibilities at the architectural level. AI-assisted language functions are intended to support intent interpretation, candidate generation, and explanation, while formal enterprise rules are intended to provide deterministic verification. Missing or low-confidence contextual information is referred to a human decision-maker rather than being assumed.

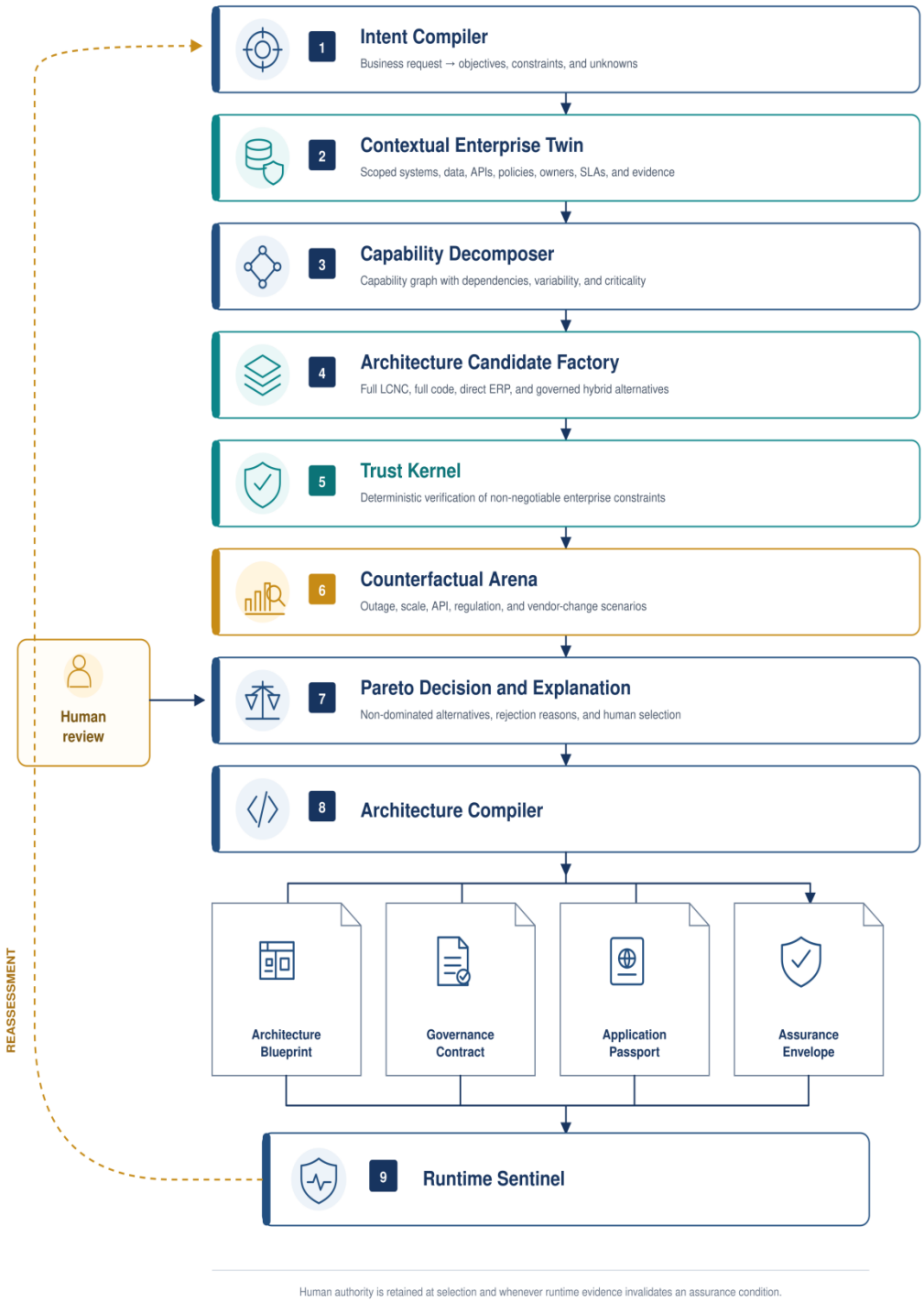

Figure 7: Conceptual reference architecture of NOVA-LCNC

The Intent Compiler translates a business request into explicit objectives, constraints and identified unknowns. The Contextual Enterprise Twin then constructs a decision-scoped model of the relevant systems, data, APIs, policies, owners, service levels, and supporting evidence. The Capability Decomposer transforms this context into a capability graph, after which the Architecture Candidate Factory generates alternative capability allocations and candidate architectures. The Trust Kernel rejects any candidate that violates formal enterprise rules. The Counterfactual Arena examines potential failures and changes in operating conditions. The Pareto Decision and Explanation module presents the feasible alternatives and explains their trade-offs to support human selection. The Architecture Compiler produces four artifacts: an Architecture Blueprint, a Governance Contract, an Application Passport, and an Assurance Envelope. Finally, the Runtime Sentinel monitors whether the assurance conditions remain valid during operation.

Table 4 summarizes the main task, inputs, outputs, and assurance controls associated with each NOVA-LCNC module. This operational description clarifies the proposed conceptual architecture and provides a foundation for future implementation. It should not be interpreted as evidence that the modules have already been implemented or extensively evaluated in practice.

NOVA-LCNC module	Main task	Input	Output	Assurance/control
Intent Compiler	Translates the business need into a structured decision problem.	Business request, process evidence, and objectives.	Intent model, constraints, unknowns.	Source traceability and confidence.
Contextual Enterprise Twin	Builds a decision-scoped model of systems, data, APIs, policies, owners, SLAs, and incidents.	Architecture repositories, policies, catalogs, operational evidence.	Scoped enterprise context model.	Provenance, date, confidence, and human query.
Capability Decomposer	Splits the process into capabilities and dependencies.	Intent model and scoped enterprise context model.	Capability graph with capabilities, dependencies, and criticality attributes.	Traceability of decomposition and dependency consistency.
Architecture Candidate Factory	Generates alternative capability allocations and candidate architectures.	Capability graph, available patterns, and platform constraints.	Candidate architectures.	Pattern diversity and feasibility constraints.
Trust Kernel	Rejects candidates that violate formal enterprise rules.	Candidate architecture and enterprise constraints.	Verified candidates and rejection reasons.	Deterministic rule evaluation and evidence traceability.
Counterfactual Arena	Evaluates plausible failures and future changes.	Verified candidates and scenarios.	Scenario impacts and required controls.	Outage, scale, API, regulation and vendor scenarios.

Pareto Decision and Explanation	Compares trade-offs without hiding them in one score.	Verified candidates and decision criteria.	Pareto frontier, explanations, and human choice.	Decision traceability.
Architecture Compiler	Compiles the selected option into architecture and governance artifacts.	Selected candidate and required controls.	Architecture Blueprint, Governance Contract, Application Passport, Assurance Envelope.	Ownership and change control.
Runtime Sentinel	Monitors assurance conditions and requests reassessment.	Runtime telemetry and Assurance Envelope.	Alerts, evidence, and reanalysis request.	Continuous assurance.

Table 4: Operational description of the conceptual NOVA-LCNC modules

NOVA-LCNC retains the central principle introduced earlier in this paper: business-side variability can be managed through NOVA-LCNC, whereas critical enterprise capabilities remain within the engineered enterprise core. A boundary is defined for each capability, evaluated through a series of architecture candidates, and captured as explicit rules and assurance conditions.

The modules can be aligned with five enterprise concerns: strategic scoping, architecture and integration, governance and security, lifecycle and DevOps, and change management. For each concern, NOVA-LCNC identifies a responsible mechanism and produces a traceable output, while leaving humans responsible for the final architecture decision.

Table 5 presents these enterprise concerns and their corresponding NOVA-LCNC mechanisms.

Enterprise dimension	Enterprise concern	NOVA-LCNC mechanism	Required output/control
Strategic scoping	Define objectives, scope, uncertainty, and relevant capabilities.	Intent Compiler + Contextual Enterprise Twin + Capability Decomposer.	Intent model, capability graph, provenance, and confidence.
Architecture and integration	Allocate capabilities and define relevant integrations.	Architecture Candidate Factory + Trust Kernel.	Verified candidates and governed API/event contracts.
Governance and security	Enforce ownership, access, auditability, compliance, publication rules.	Trust Kernel + Architecture Compiler.	Governance Contract and Application Passport.
Lifecycle and DevOps	Test, deploy, observe, rollback, maintain, reassess the solution.	Counterfactual Arena + Runtime Sentinel.	Assurance Envelope, telemetry, and reanalysis trigger.
Change management	Preserve accountability, readiness, training, and escalation.	Pareto Decision and Explanation + human approval.	Traceable decision, named owner, and escalation path.

Table 5: Core enterprise concerns and corresponding NOVA-LCNC mechanisms

At this preliminary stage, NOVA-LCNC can be represented using a lightweight conceptual metamodel. Its main concepts are Business Intent, Context Fact, Evidence Source, Capability, Dependency, Candidate

Architecture, Constraint, Scenario, Decision, Governance Control, and Assurance Condition. Where applicable, each Context Fact retains its source, date, and confidence level. Candidate Architectures retain verification results and reasons for rejection, while the selected architecture records the conditions under which it remains acceptable.

i. Neuro-symbolic decision and assurance boundary

The distribution of responsibilities is made clear in Figure 8 through AI-assisted functions, a symbolic Trust Kernel, and human decision-making.

AI-powered functions analyze business intent, extract the relevant context within the defined scope, create architecture alternatives and counterfactual scenarios, and explain the trade-offs. The symbolic Trust Kernel verifies these alternatives against enterprise requirements and provides clear evidence for acceptance or rejection. Human experts validate the scope, resolve uncertainties, approve the criteria and scenarios, and select an acceptable alternative.

NOVA-LCNC therefore does not propose solving the problem by reducing it to a single choice: code or LCNC. It breaks the process down into capabilities, generates several architecture alternatives, discards those that do not comply with enterprise rules, positions the remaining alternatives on a Pareto frontier, and records the conditions under which the selected architecture remains valid in an Assurance Envelope.

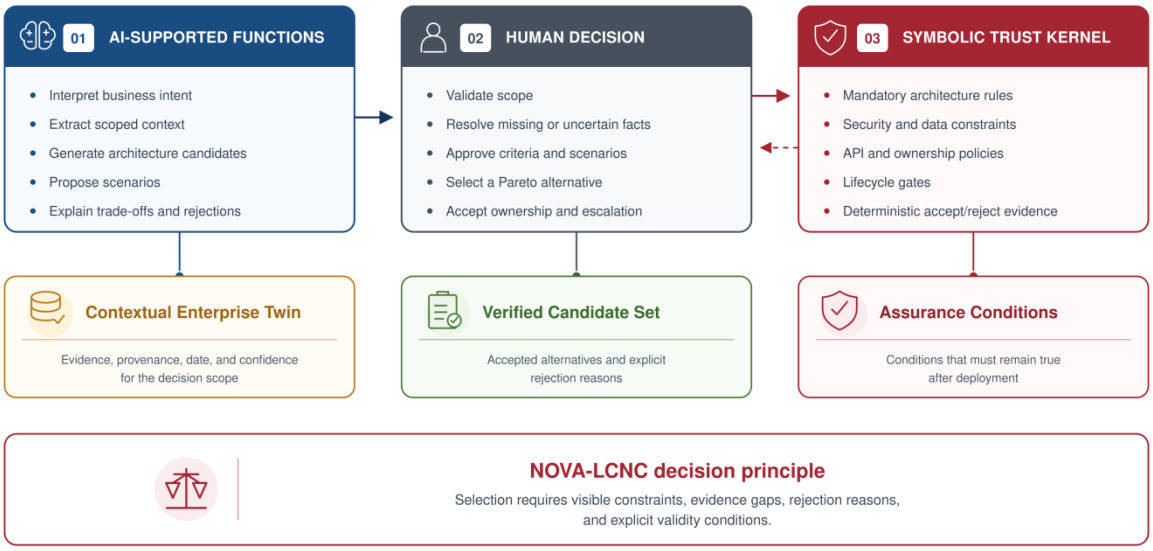

Figure 8: Neuro-symbolic decision and assurance boundary in NOVA-LCNC

j. Illustrative use case: applying NOVA-LCNC to an ERP-Integrated Purchase Request and Budget-Control Workflow

This subsection covers the digitalization of an internal purchase request workflow using NOVA-LCNC. The use case does not provide empirical validation. It is intended to demonstrate capability decomposition, candidate generation, rule-based verification, counterfactual analysis, architecture selection, governance compilation, and continuous assurance in a realistic enterprise context.

From an enterprise information systems perspective, the purchase request workflow involves supplier master data, budget availability, approval criteria, accounting rules, user authorizations, audit requirements, and ERP transactions. NOVA-LCNC makes the related evidence, ownership, interfaces, constraints, and assurance conditions explicit before an architecture is selected.

The Intent Compiler captures the aim of the digitalization initiative. The Contextual Enterprise Twin captures the ERP, supplier master data, budget service, IAM, API catalogue, owners, service levels, and policies required to make the decision. The Capability Decomposer identifies the request form, approval process, notifications, supplier search, budget verification, accounting rules, ERP posting, and audit capabilities. Using this capability graph, the Architecture Candidate Factory can produce several complete alternatives without immediately committing to a single application.

Figure 9 shows how the NOVA-LCNC pipeline can be applied to this use case through four candidate architectures: a full LCNC solution, a direct integration between LCNC and the ERP, a full-code solution, and a governed hybrid involving registered APIs, central services, and events. The first two are rejected by the Trust Kernel because they violate master-data consistency, transaction, or integration constraints. Although the full-code

candidate remains viable, the governed hybrid provides a better balance of agility, control, and portability in this illustrative case. The final selection remains a human decision.

Figure 9: Illustrative application of NOVA-LCNC to an ERP-integrated purchase request and budget-control workflow

The selected governed hybrid places the request form, approval workflow, dashboard, and notifications within the LCNC platform; exposes supplier data through a governed read-only API; keeps budget validation, accounting rules, and ERP posting within authorized enterprise services; and uses IAM, events, audit, and

observability across the integration boundary, as illustrated in Figure 9. When ERP unavailability is considered, the counterfactual analysis introduces a pending mode, queues, retries, idempotency, and compensation mechanisms.

NOVA-LCNC element	Application in the use case	Result
Intent Compiler	Digitalize purchase requests and approvals while preserving financial and ERP controls.	Process-oriented need.
Contextual Enterprise Twin	ERP, supplier master, budget service, IAM, API catalog, owners, SLAs, and policies.	Decision-scoped evidence model with provenance and confidence.
Capability Decomposer	Form, notification, approval, supplier lookup, budget check, accounting, ERP posting, and audit.	Capability graph with dependencies and criticality.
Architecture Candidate Factory	Full LCNC, direct LCNC-ERP, full code, and governed hybrid.	Four architecture alternatives.
Trust Kernel	Applies master-data, transaction, IAM, audit, ownership, lifecycle, and API rules.	Full LCNC and direct LCNC-ERP candidates rejected.
Counterfactual Arena	ERP outage, volume x 10, API or regulation change, and platform exit.	Queue, retry, idempotence, compensation, capacity, and portability controls.
Pareto Decision and Explanation	Compares agility, control, cost, portability, maintainability, and operational risk.	Governed hybrid selected by the accountable human decision-maker.
Architecture Blueprint	LCNC form/workflow/dashboard, governed APIs, budget service, events, IAM, audit, and ERP.	Target component, interface, and data-flow architecture.
Governance Contract	API allowlist, access rules, approval thresholds, audit obligations, owners, and deployment.	Traceable control specification.
Application Passport	Purpose, owner, systems, data, dependencies, criticality, and lifecycle status.	Application traceability and accountability.
Assurance Envelope	Volume ceiling, ERP API latency and availability, data classification, owner, regulations, and API version.	Explicit validity conditions for the selected architecture.
Runtime Sentinel	Monitors telemetry, policy drift, API drift, and ownership status.	Alert and NOVA-LCNC reassessment request when a condition fails.

Table 6: Application of NOVA-LCNC to the ERP-integrated purchase request and budget-control use case

The selected architecture is not determined simply by choosing the lowest or highest score. NOVA-LCNC first eliminates candidates that violate enterprise rules and then ranks the remaining candidates according to criteria such as agility, control, portability, maintainability, cost, and operational risk. The final choice therefore remains a human decision, supported by clear reasons, required controls, and a traceability record for the selected hybrid.

The use case also illustrates the role of the Assurance Envelope. The architecture remains acceptable as long as the stated conditions are met, including transaction-volume limits, required ERP API availability and latency, data classification, a named application owner, applicable regulations, and valid registered API contracts. When one of these conditions changes, the Runtime Sentinel no longer assumes that the original decision remains valid and requests a reassessment.

k. Limited empirical grounding in complex enterprise settings

The corpus contains empirical studies, expert panels, datasets of practitioners, and some no-code application cases [3], [5], [7], [10], [12], [19], [25], but the empirical basis for LCNC integration in complex enterprise settings remains limited. Specifically, there is a lack of longitudinal studies or multiple case studies showing how LCNC integration evolves in real companies.

This indicates that many frameworks remain more conceptual than practical. They provide helpful theoretical insights but remain insufficiently tested in large and complex enterprise settings.

l. Feasibility, transition and change management

It has been well documented in the literature that feasibility and change management are issues in the integration of LCNC. The process is not only a technical challenge, but an organizational change as well. It includes issues such as training, legitimacy, resistance to change, changing roles and responsibilities, and the need to balance business autonomy with IT governance [9], [12], [25], [29].

These challenges are further exacerbated by the emergence of no-code and LLM-driven no-code solutions [20]-[24] as these solutions enable easier development and introduce new complexities. Therefore, the adoption of LCNC must be accompanied by transition strategies that will support the organization over the long term, not only through technical solutions.

m. Practical implications

In practice, the five enterprise concerns identified in the review still need to be addressed, but NOVA-LCNC links them to mechanisms that make them traceable and easier to understand. Strategic scoping is supported by the Intent Compiler, Contextual Enterprise Twin, and Capability Decomposer; architecture and integration by the Architecture Candidate Factory and Trust Kernel; governance and security by formal rules and the Governance Contract; organizational accountability by Pareto-based explanations, explicit ownership, and human approval; and lifecycle and DevOps by the Runtime Sentinel and Assurance Envelope.

i. Trust Kernel and executable governance controls

In practice, LCNC can add value by helping teams develop interfaces, workflows, dashboards and simple business applications faster. However, a seemingly trivial application on the surface may slowly develop internal complexity within the information system.

This underlying complexity can manifest itself in several ways, such as duplicated data, unclear ownership, weak access control, poorly documented workflows, fragile connectors, or missing validation steps.

NOVA-LCNC converts these recurring risks into explicit constraints before deployment. The Trust Kernel links each risk to a verifiable rule, a rejection reason and a control that can be compiled into the Governance Contract. Figure 10 illustrates this risk-to-rule-to-control transformation.

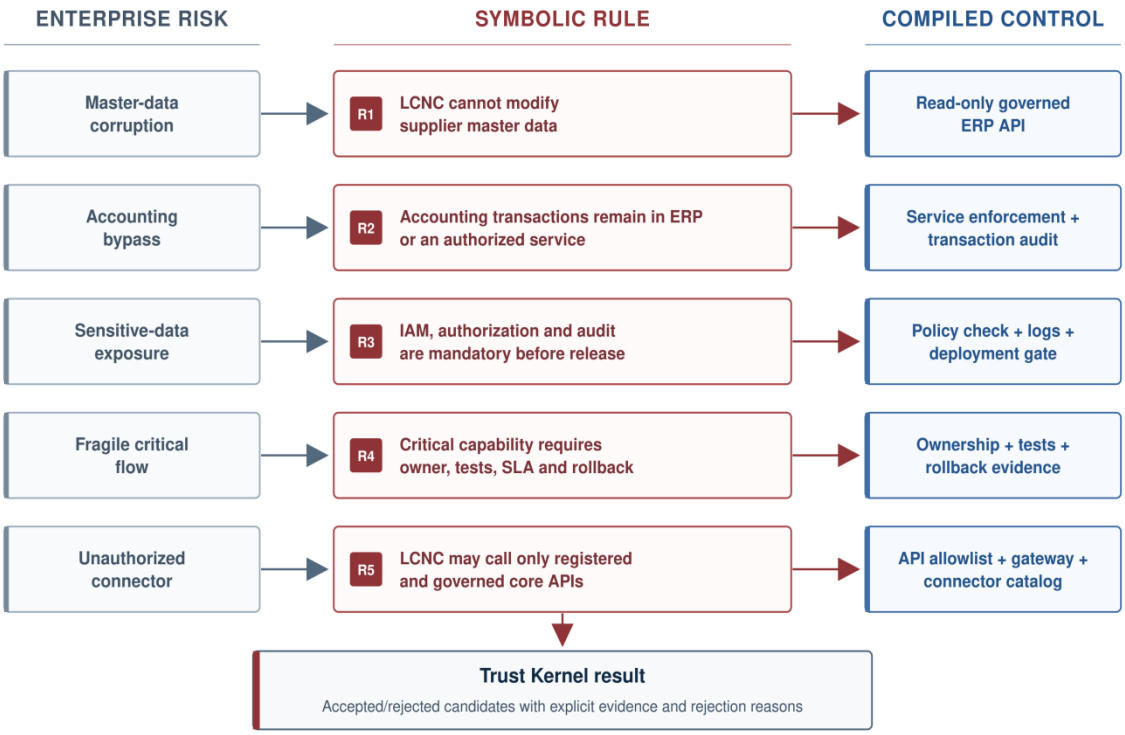

Figure 10: NOVA-LCNC Trust Kernel risk-to-rule-to-control mapping

Each LCNC initiative should be managed by an application registry to reduce hidden complexity, a clear ownership model, publication approval processes, integration through an API-first approach, a connector catalog, periodic performance reviews, and technology rationalization. This helps to ensure solution tracking, reuse, and maintenance responsibilities. Governance should be designed as an integral component, enabling LCNC adoption while maintaining security, traceability, reusability, and maintainability at enterprise scale.

n. Managerial and practical implications

For instance, NOVA-LCNC could prevent LCNC applications from editing supplier master data, require accounting transactions to remain within the ERP or an authorized service, and require capabilities to be subject to IAM and audit controls before publication. For critical capabilities, it could also require named owners, tests, service levels, and rollback procedures. In addition, only registered core APIs would be available for use by LCNC applications. These obligations are documented in the resulting Governance Contract, alongside the Application Passport and Architecture Blueprint.

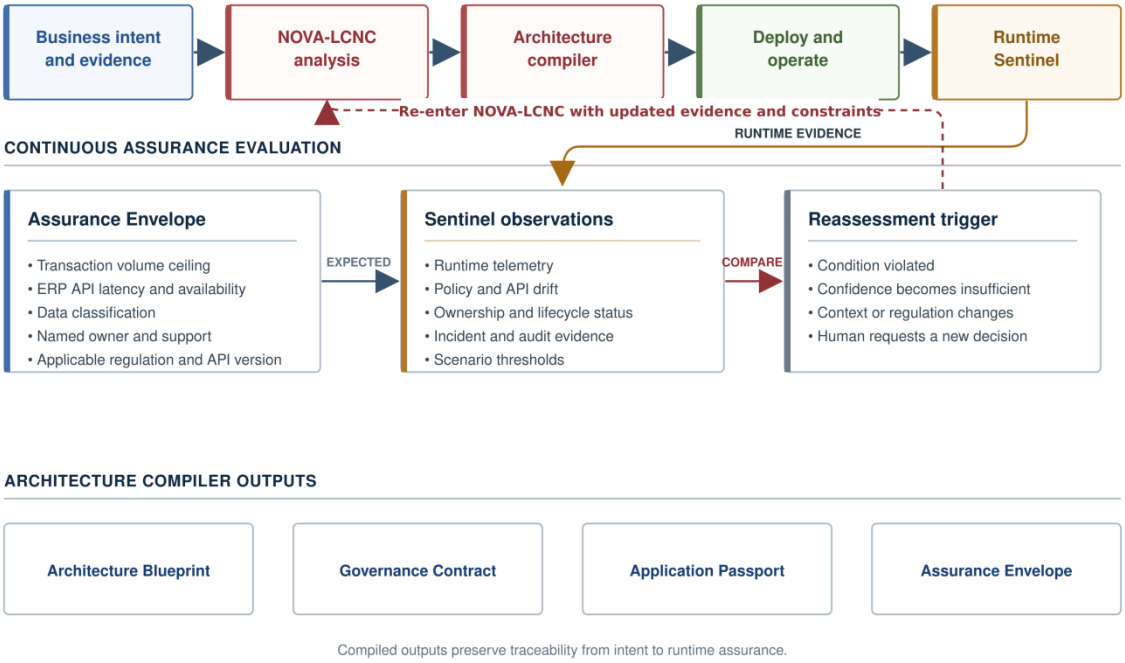

Figure 11: NOVA-LCNC continuous assurance lifecycle

From the perspective of managers and enterprise architects, NOVA-LCNC follows the same central principle: governance should be designed from the very beginning. It provides a traceable progression from scoping evidence and capability decomposition to the comparison of alternatives and verification of mandatory controls, while final authority remains with accountable human decision-makers. The resulting architecture is then used for continuous assurance, as summarized in Figure 11.

V. Research agenda

a. Research direction 1: implementation, validation, and refinement of NOVA-LCNC

Future research should implement, test, and optimize NOVA-LCNC through a prototype, expert reviews, enterprise case studies, and comparative evaluations. The prototype should support scoped context and provenance, capability graphs, formal constraints, reusable candidate patterns, Pareto-based explanations, compiled governance artifacts, and assurance monitoring. Decision quality, rule coverage, the explanatory value of the rules, required effort, scalability, and organizational acceptability should also be analyzed.

b. Research direction 2: differentiating low-code and no-code in enterprise use

The literature still tends to treat low-code and no-code as a single category, failing to provide insight into the specific functions of each in an enterprise environment. For future research, they should be clearly differentiated in their strengths, limitations and use cases, with consideration of governance requirements, scalability, maintainability, and organizational needs. This differentiation should clarify when each is more suitable.

c. Research direction 3: interoperability and lock-in assessment

Interoperability and vendor lock-in remain significant barriers to LCNC adoption. Future work will focus on operationalizing portability evidence within the Contextual Enterprise Twin and evaluating this evidence in the Counterfactual Arena using portability test scenarios. This would enable organizations to identify migration risks and determine the conditions under which a selected architecture should be reevaluated.

d. Research direction 4: governance models for citizen development and no-code growth

As more citizen-development rules are implemented across no-code platforms, future studies could assess how the NOVA-LCNC Trust Kernel and Governance Contract can translate these rules into enforceable controls while preserving human accountability. This includes role formalization, publication gates, API allowlists, security controls, evidence requirements, escalation paths, and exception handling.

e. Research direction 5: lifecycle and DevOps practices

Future work is needed on the Assurance Envelope and Runtime Sentinel to address the lifecycle dimension of NOVA-LCNC. Testing and versioning, deployment and observability, rollback and maintenance, ownership, API drift, and policy drift should be linked to explicit architecture validation conditions rather than treated as post-deployment tasks.

Future studies should also explore how low-code, no-code, and AI-assisted no-code applications can interact with enterprise DevOps and observability pipelines to automate the review process and, where necessary, trigger a new NOVA-LCNC analysis based on runtime evidence.

VI. Limitations

This study has several limitations. First, NOVA-LCNC remains a conceptual reference architecture and has not yet been implemented or empirically validated. Therefore, the feasibility, completeness, scalability, and effectiveness of the architecture and its components, including its modules, rules, candidate-generation mechanisms, and Runtime Sentinel, have not been tested. Second, the corpus of thirty publications provides a balanced representation of low-code and no-code research, but the field remains young. Research on low-code is more mature, while studies of no-code and AI-assisted no-code remain largely exploratory. Third, the review is less exhaustive than a full systematic literature review and does not include a meta-analysis.

These limitations point to several directions for future research: developing an architectural prototype, formalizing and testing the initial Trust Kernel rules, comparing the generated alternatives and explanations with expert assessments, and evaluating the Assurance Envelope in real enterprise scenarios.

VII. Conclusion

Thirty publications were critically reviewed through a structured thematic approach to explore how low-code/no-code platforms are integrated into complex enterprise information systems. The analysis revealed that LCNC platforms provide numerous benefits with respect to development speed, accessibility, business-IT collaboration and democratized application development. At the same time, their sustainable use in enterprise settings remains constrained by several challenges, including software quality expectations, role transformation, governance needs, limitations in interoperability and architectural complexity.

The review also showed that the field is active, but quite fragmented, with the low-code literature being more prominent in enterprise integration, interoperability limitations, governance and lifecycle studies, and the no-code literature being more prominent in accessibility studies, product development studies, and AI-assisted generation studies. A key conclusion of this article is that a central way to go beyond individual platform categories is to shift to more structured and flexible enterprise integration frameworks.

In this respect, the reviewed corpus highlights an important scientific gap: the limited availability of approaches that combine adoption, architecture, interoperability, governance, software quality, and organizational transition within a unified LCNC-EIS integration framework.

To address these gaps, the paper outlines a research agenda covering five areas: the implementation and validation of NOVA-LCNC, a clearer distinction between low-code and no-code, interoperability and lock-in assessment, operational governance mechanisms for citizen development, and continuous assurance based on lifecycle and DevOps practices.

Finally, sustainable LCNC integration is not only about speed and accessibility. The main contribution of this paper is the proposal of an initial neuro-symbolic reference architecture, NOVA-LCNC, designed to decompose business needs into capabilities, synthesize and verify LCNC-EIS architecture alternatives, expose trade-offs for human decision-making, compile governance and assurance artifacts, and define conditions for reassessment. Since NOVA-LCNC has not yet been implemented or empirically evaluated, implementation and validation represent important next steps in this doctoral research.

References

[1] M. O. Ajimati, N. Carroll, and M. Maher, "Adoption of low-code and no-code development: A systematic literature review and future research agenda," Journal of Systems and Software, vol. 222, Article 112300, 2025, doi: 10.1016/j.jss.2024.112300.

[2] S. Käss, S. Strahringer, and M. Westner, "Drivers and inhibitors of low code development platform adoption," in Proceedings of the 2022 IEEE 24th Conference on Business Informatics (CBI), vol. 1, pp. 196–205, 2022, doi: 10.1109/CBI54897.2022.00028.

[3] S. Käss, S. Strahringer, and M. Westner, "Practitioners' perceptions on the adoption of low code development platforms," IEEE Access, vol. 11, pp. 29009–29034, 2023, doi: 10.1109/ACCESS.2023.3258539.
[4] R. Domański, H. Wojciechowski, J. Lewandowicz, and Ł. Hadaś, "Digitalization of management processes in small and medium-sized enterprises: An overview of low-code and no-code platforms," Applied Sciences, vol. 13, no. 24, Article 13078, 2023, doi: 10.3390/app132413078.
[5] M. A. A. Al Alamin, G. Uddin, S. Malakar, S. Afroz, T. B. Haider, and A. Iqbal, "Developer discussion topics on the adoption and barriers of low code software development platforms," Empirical Software Engineering, vol. 28, Article 4, 2023, doi: 10.1007/s10664-022-10244-0.
[6] D. Pinho, A. Aguiar, and V. Amaral, "What about the usability in low-code platforms? A systematic literature review," Journal of Computer Languages, vol. 74, Article 101185, 2023, doi: 10.1016/j.cola.2022.101185.
[7] T. Guthardt, J. Kosiol, and O. Hohlfeld, "Low-code vs. the developer: An empirical study on the developer experience and efficiency of a no-code platform," in Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems Companion (MODELS Companion '24), pp. 856–865, 2024, doi: 10.1145/3652620.3688332.
[8] M. Moskal, "No-code application development on the example of Logotec App Studio platform," Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska, vol. 11, no. 1, pp. 54–57, 2021, doi: 10.35784/iapgos.2429.
[9] G. Masili, "No-code development platforms: Breaking the boundaries between IT and business experts," International Journal of Economic Behavior, vol. 13, no. 1, pp. 33–49, 2023, doi: 10.14276/2285-0430.3705.
[10] S. Heuschkel, "The impact of no-code on digital product development," arXiv, 2023, doi: 10.48550/arXiv.2307.16717.
[11] K. Laeeq, "Drag & Build: Democratizing app development through a no-code drag-and-drop platform," Societal Transformation: AI and Big Data, vol. 3, no. 1, Article 17, 2025.
[12] B. Binzer, E. Elshan, D. Fürstenau, and T. J. Winkler, "Establishing a low-code/no-code-enabled citizen development strategy," MIS Quarterly Executive, vol. 23, no. 3, pp. 253–273, 2024.
[13] S. Iho, D. Krejci, and S. Missonier, "Supporting knowledge integration with low-code development platforms," in ECIS 2021 Research Papers, Paper 38, 2021.
[14] A. Sahay, A. Indamutsa, D. Di Ruscio, and A. Pierantonio, "Supporting the understanding and comparison of low-code development platforms," in Proceedings of the 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 171–178, 2020, doi: 10.1109/SEAA51224.2020.00036.
[15] I. Alfonso, A. Conrardy, and J. Cabot, "Towards the interoperability of low-code platforms," arXiv, 2024, doi: 10.48550/arXiv.2412.05075.
[16] A. Lamanna, "A structured evaluation framework for low-code platform selection: A multi-criteria decision model for enterprise digital transformation," arXiv, 2025.
[17] A. Bucaioni, A. Cicchetti, and F. Ciccozzi, "Modelling in low-code development: A multi-vocal systematic review," Software and Systems Modeling, vol. 21, pp. 1959–1981, 2022, doi: 10.1007/s10270-021-00964-0.
[18] D. Di Ruscio, D. S. Kolovos, J. de Lara, A. Pierantonio, M. Tisi, and M. Wimmer, "Low-code development and model-driven engineering: Two sides of the same coin?," Software and Systems Modeling, vol. 21, pp. 437–446, 2022, doi: 10.1007/s10270-021-00970-2.
[19] S. Rafi, M. A. Akbar, M. Sánchez-Gordón, and R. Colomo-Palacios, "DevOps practitioners' perceptions of the low-code trend," in Proceedings of the 16th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), pp. 301–306, 2022, doi: 10.1145/3544902.3546635.
[20] M. Monteiro, B. Castelo Branco, S. Silvestre, G. Avelino, and M. T. Valente, "NoCodeGPT: A no-code interface for building web apps with language models," Software: Practice and Experience, vol. 55, no. 8, pp. 1408–1424, 2025, doi: 10.1002/spe.3432.
[21] R. Divya, S. Kavidarshini, P. Santhosh, and S. Shashank, "SmartAPIForge: A no-code platform for automated REST API generation from natural language," International Journal of Advanced Research in Computer and Communication Engineering, vol. 14, no. 12, 2025, doi: 10.17148/IJARCCE.2025.141262.
[22] M. Wang, T. Pfandzelter, T. Schirmer, and D. Bermbach, "LLM4FaaS: No-code application development using large language models and Function-as-a-Service," arXiv, 2025.
[23] H. An, Y. Kim, W. Seo, J. Park, D. Kang, C. Oh, D. Kim, and S. Lee, "AIAP: A no-code workflow builder for non-experts with natural language and multi-agent collaboration," arXiv, 2025.
[24] P. Pattnayak and H. Bohra, "Review of tools for zero-code LLM based application development," arXiv, 2025, doi: 10.48550/arXiv.2510.19747.
[25] A. Viljoen, M. Radić, A. Hein, J. Nguyen, and H. Krcmar, "Governing citizen development to address low-code platform challenges," MIS Quarterly Executive, vol. 23, no. 3, pp. 305–324, 2024, doi: 10.17705/2msqe.00100.

[26] E. Martinez and L. Pfister, "Benefits and limitations of using low-code development to support digitalization in the construction industry," Automation in Construction, vol. 152, Article 104909, 2023, doi: 10.1016/j.autcon.2023.104909.
[27] G. F. Hurlburt, "Low-code, no-code: What's under the hood?," IT Professional, vol. 23, no. 6, pp. 4–7, 2021, doi: 10.1109/MITP.2021.3123415.
[28] J. Cui, "The impact of low-code and no-code programming on software product delivery quality and development efficiency," SSRN, 2024, doi: 10.2139/ssrn.5010766.
[29] J.-M. Mottu and G. Sunyé, "Emerging new roles for low-code software development platforms," in Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems Companion (MODELS Companion '24), pp. 905–914, 2024, doi: 10.1145/3652620.3688337.
[30] B. Naqvi, D. Kedziora, L. Zhang, and S. Oyedeji, "Quality of low-code/no-code development platforms through the lens of ISO/IEC 25010:2023," Computer, vol. 58, no. 3, pp. 30–40, 2025.