From Knowledge Base to Intentional Framework: Evolving a Low-Code Platform Through Interactive UI Design and AI Integrations

Andrea ALBERICI[1], Nevila BACI[2], Eugenio BRENTARI[3], Kozeta SEVRANI[4]

Abstract

This paper presents findings from a multi-year action design research project investigating the development of a low-code platform through an intensional framework approach. While the initial research deliverable focused on creating a comprehensive knowledge base for Domain-Specific Languages (DSLs), our investigation revealed that the interactive user interface layer, implemented through a specific javascript application, could serve as an intentional framework for the entire low-code platform. The research employed a combination of Design Thinking and Action Design Research methodologies, analyzing over 5,000 cases across multiple business sectors.

Our findings demonstrate that the presentation layer's ability to handle complexity and taxonomy, combined with its dynamic visualization capabilities, created an emergent intensional framework that surpassed the original knowledge base objectives. The implementation of this framework through the specific ajax application provided a self-contained, single-page web application that effectively managed the intensional relationships between different components of the low-code platform.

The research identified several key advantages of this approach, including enhanced comprehension of complex software artifacts, improved navigation of relationships between components, and increased flexibility in representing business rules and domain-specific constraints.

[1] Department of Statistics and Applied Informatics, Faculty of Economy, University of Tirana, Albania; andrea.alberici@unitir.edu.al
[2] Department of Statistics and Applied Informatics, Faculty of Economy, University of Tirana, Albania; nevila.baci@unitir.edu.al
[3] Faculty of Economy, Università degli Studi di Brescia, Italy; eugenio.brentari@unibs.it
[4] Department of Statistics and Applied Informatics, Faculty of Economy, University of Tirana, Albania; kozeta.sevrani@unitir.edu.al

However, limitations were also observed, particularly in areas of scalability and offline synchronization.
Looking ahead, our research roadmap focuses on integrating Generative AI and Agentic Retrieval-Augmented Generation (RAG) into the framework. This integration aims to enhance the platform's capability to understand context, generate appropriate code segments, and provide more intelligent assistance in low-code development processes. These advancements are expected to create a more robust and adaptive intensional framework that can better serve the evolving needs of modern software development.

Keywords: Intensional Framework, Low-Code Platform, Domain-Specific Languages, Action Design Research, Interactive UI, Generative AI, RAG, Knowledge Base.

1. Introduction

The sector of software development has undergone remarkable transformation with the rise of low-code platforms aimed at expediting development timelines while minimizing the need for extensive technical skills. These platforms try to make software development accessible to individuals with diverse levels of technical expertise, allowing them to create intricate applications using visual tools and pre-existing components. Nonetheless, as organizations increasingly depend on these platforms for critical applications, the demands for complexity and sophistication have escalated significantly.

Our research collects these emerging challenges by introducing an innovative approach that redefines the conventional knowledge base used at the core for the creation of low-code platforms as an intensional framework. This change signifies a crucial shift from static information repositories to dynamic, interconnected systems that adaptively factor in user requirements and contextual factors. The intensional framework methodology offers a more versatile, scalable, and responsive foundation for low-code development compared to traditional approaches.

This paper outlines the results of our multi-year action design research initiative, which utilized a blend of Design Thinking and Action Design Research techniques. Through a methodical process involving problem identification, ideation, prototyping, and testing, coupled with cycles of

building, intervention, evaluation, and reflection, we crafted and enhanced an intensional framework that significantly improves the functionalities of low-code platforms.

The structure of this paper is organized as follows:

Section 2 examines pertinent literature concerning low-code platforms, intensional frameworks, and domain-specific languages; Section 3 elaborates on our research methodology; Sections 4, 5, and 6 detail, respectively, the development of our knowledge base, the creation of an interactive user interface, and the transition to an intensional framework. Section 7 elaborates on projections for the future, particularly highlighting the role of AI integration, and Section 8 finishes the document with summarizing insights about the ramifications and potential stemming from our research.

2. Literature Review

Our literature review analyzes foundational research across four key domains relevant to our intensional framework approach. We explore the evolution of low-code development platforms that have reshaped software engineering through visual interfaces and pre-built components. We investigate Domain-Specific Languages (DSLs) as tailored programming languages for specific domains, linking technical implementation with business needs. The review also covers intensional computing and frameworks, differentiating between extensional systems that define elements by enumeration and intensional systems that define them through properties and relationships. Lastly, we assess research on interactive user interfaces for complex systems, emphasizing methods that improve understanding and exploration of complex information structures.

2.1 Low-Code Development Platforms

Low-code development platforms have surfaced as an influential trend within the realm of software engineering, facilitating swift application development with minimal hand-coding (Waszkowski, 2019). Generally, these platforms showcase visual interfaces, pre-configured components, and drag-and-drop options that facilitate users in crafting applications without the necessity of extensive programming skills (Vincent et al., 2019). The market for low-code development platforms has witnessed considerable expansion, with Forrester forecasting that by 2022, the market will reach $21.2 billion (Rymer et al., 2017).

Investigations into low-code platforms have concentrated on several critical areas, such as their efficacy in speeding up development cycles (Sahay et al., 2020), their function in democratizing software development (Bock & Frank, 2021), and the obstacles they encounter within enterprise settings (Sanchis et al., 2020). Despite their benefits, low-code platforms frequently grapple with challenges related to complexity management, extensibility, and integration with existing systems (Tisi et al., 2019).

2.2 Domain-Specific Languages (DSLs)

Domain-Specific Languages are specialized programming languages crafted for particular problem areas, providing suitable abstractions and notations (van Deursen et al., 2000). DSLs improve productivity, reliability, and maintainability by encapsulating domain knowledge and constraints (Mernik et al., 2005). They act as a conduit between domain experts and software developers, enhancing communication and collaboration (Kelly & Tolvanen, 2008).

Within the framework of low-code platforms, DSLs are pivotal in augmenting the platform's functionalities and tailoring it for specific domains (Völter, 2013). They allow for the articulation of intricate domain rules and constraints in a manner that is both machine-executable and comprehensible to humans (Kosar et al., 2016). However, the creation and upkeep of DSLs often demand considerable expertise and resources, posing challenges to their broader implementation (Fowler, 2010).

2.3 Intensional Computing and Frameworks

Intensional computing denotes a paradigm in which a system's behavior is influenced not only by the explicit data and instructions but also by context and intentions (Wadge & Ashcroft, 1985). In contrast to extensional systems that define elements through enumeration, intensional systems characterize elements by their properties and relationships (Orchard, 2014).

Intensional frameworks expand upon this concept to provide structures that adjust according to context and intent, offering enhanced flexibility and expressiveness compared to traditional systems (Plasmeijer & van Eekelen, 1993). These frameworks have been utilized in diverse fields, including natural language processing (Blackburn & Meyer-Viol, 1994), database systems (Buneman et al., 1995), and programming languages (Wadler, 1992).

In software engineering, the investigation of intensional frameworks has been conducted to advance abstraction, modularity, and adaptability (Kiczales et al., 1997). They offer mechanisms for defining and manipulating relationships among components, enabling more advanced interaction patterns (Orchard & Schrijvers, 2010). That said, inquiries into the integration of intensional frameworks in low-code platforms are still few, pointing to an important absence in the scholarly work that our research aspires to cover.

2.4 Interactive User Interfaces for Complex Systems

Interactive user interfaces designed for complex systems aim to enhance user experience and understanding through visual representation, interaction design, and information architecture (Yi et al., 2007). Successful interfaces for complex systems should aid cognitive processes, minimize cognitive load, and promote the understanding of intricate relationships (Shneiderman & Plaisant, 2010).

Analyses in this realm have scrutinized multiple methodologies, which include interactive visual tools (Heer & Shneiderman, 2012), versatile interfaces (Jameson, 2009), and contextually aware designs (Dey, 2001). These strategies seek to connect the complex underlying systems with user understanding, thereby enabling more effective interaction and management (Card et al., 1999).

Within the low-code platforms, interactive user interfaces are essential for making intricate development tasks approachable for users with diverse levels of technical skills (Luo et al., 2020). Nevertheless, creating interfaces that accurately depict complex relationships and dependencies while ensuring usability continues to pose a considerable challenge (Barricelli et al., 2019).

3. Research Methodology

Our research employed an innovative and hybrid methodology that combined the principles of Design Thinking and Action Design Research (ADR) approaches, resulting in a multifaceted framework that was particularly effective for comprehensively addressing the intricate socio-technical problems we encountered. This integrated methodology not only facilitated user-centered innovation by placing the end user at the forefront of our design process but also ensured the implementation of rigorous, intervention-based research practices that contributed to our understanding of the domain.

3.1 Research Approach: Integrated Design Thinking and Action Design Research

In the course of our research, we adopted an integrated and synergistic approach that seamlessly merged the methodologies of Design Thinking with those of Action Design Research (ADR), all aimed at effectively addressing the complex challenge associated with the development of an extensive business knowledge base specifically tailored for a low-code platform that utilizes Domain-Specific Languages (DSLs). This integration turned out to be particularly efficient in systematically cataloging, classifying, and establishing connections among over 5000 real-world use cases that spanned across a multitude of diverse business domains, thereby enhancing the richness of our findings.

The Design Thinking component served as the guiding force behind our user-centered innovation process, which commenced with a meticulous phase of problem identification during which we uncovered and highlighted significant gaps in the ways that domain knowledge was being stored, accessed, and utilized within existing platforms in the market. Through a series of collaborative ideation sessions that included domain experts, software developers, and business analysts, we successfully conceptualized a structured and organized knowledge repository designed to capture the intricate nuances of business rules and processes via the implementation of Domain-Specific Languages (DSLs). Our prototyping efforts evolved significantly, transitioning from abstract conceptual taxonomies and classification systems to more concrete and tangible representations of real-world use cases complete with their associated DSL implementations, thereby enhancing clarity and usability. Engaging in user testing with actual business stakeholders provided us with critical feedback regarding the utility, comprehensibility, and applicability of our approach, which led to iterative refinements of both the overarching knowledge structure and the various methods of presentation employed.

In parallel to this, the Action Design Research methodology provided us with the essential rigor and methodological framework necessary for the successful implementation and thorough evaluation of this extensive knowledge base within authentic organizational contexts that reflect real-world scenarios. During the building phase, we meticulously developed a robust and comprehensive framework capable of efficiently storing, categorizing, and relating the use cases in accordance with our evolved taxonomy, which included various dimensions such as business sectors, types of modularity,

categories of DSLs, and specific application domains. The subsequent intervention phase involved the practical deployment of this meticulously constructed knowledge base that spanned various sectors, thereby enabling business users to readily access and effectively apply the cataloged solutions to their unique and specific challenges encountered in their operational environments. Through a systematic and thorough evaluation process that combined usage analytics, solution effectiveness metrics, and in-depth interviews with users, we gathered a wealth of comprehensive data regarding how the knowledge base functioned both as a repository of information and as a practical problem-solving tool.

The recursive phases of analysis allowed us to transform these real-world practical experiences into valuable theoretical insights concerning how domain knowledge can be effectively structured, represented, and applied within low-code environments, which directly informed subsequent iterations and ultimately facilitated the evolution toward an intensional framework approach. This dual methodology empowered us to maintain a delicate balance between practical relevance and theoretical contribution throughout the entirety of the research process, which was essential for successfully navigating a project of such considerable scale and inherent complexity.

4. Phase 1: Knowledge Base Development

The initial phase of our research focused on developing a comprehensive knowledge base for Domain-Specific Languages (DSLs) that could serve as a foundation for low-code platform development. This phase, conducted from 2016 to 2018, followed a systematic approach to identify, collect, and organize solutions to generic business problems across multiple sectors.

4.1 Problem Identification and Scoping

Our investigation commenced with a thorough examination of the obstacles organizations encounter when attempting to implement and leverage DSLs for solving business challenges. Through interviews with stakeholders, a review of existing literature, and analysis of industry practices, we pinpointed several critical issues:

1. Absence of Centralized Knowledge: Organizations often found themselves reinventing solutions for recurring issues due to a lack of a centralized repository for DSL-based solutions. Restricted Accessibility: Frequently, existing DSL solutions were stored in formats that obstructed intuitive understanding and exploration.

Inefficient Knowledge Transfer: The lack of standardized methods for documenting and disseminating DSL solutions obstructed collaboration and the transfer of knowledge. Cumbersome Navigation: The process of sifting through extensive collections of cases was tedious, hindering the extraction of insights and the identification of patterns.

2. In light of these findings, we delineated the focus of our research to encompass:
3. Identifying common business challenges across relevant industries Gathering and documenting solutions utilizing DSLs Creating a structured knowledge base to systematically organize these solutions Developing a presentation layer to improve accessibility and exploration

4.2 Literature Review and Knowledge Base Development

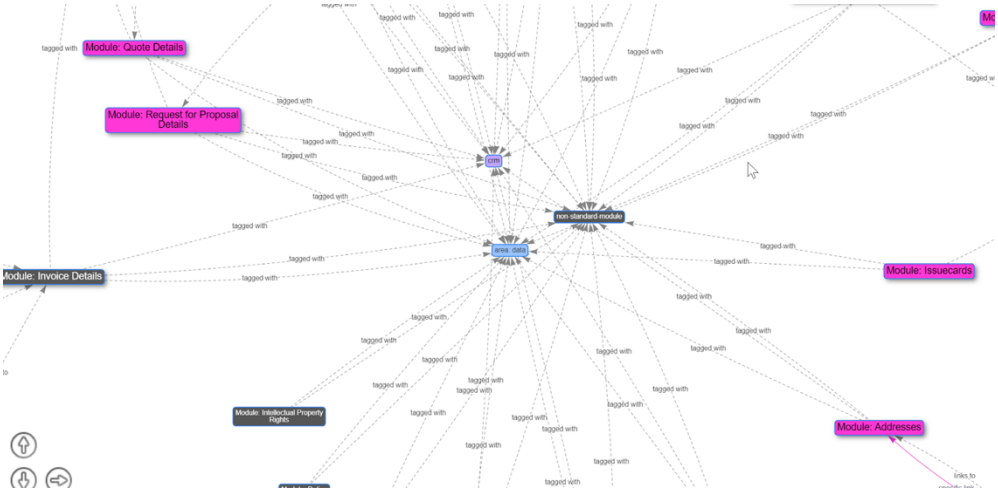
The theoretical framework for our knowledge base emerged from a comprehensive interdisciplinary literature review, which explored research in Domain-Specific Languages, knowledge management systems, business process modeling, and human-computer interaction fields. This theoretical underpinning shaped our practical methodology for designing and constructing a robust knowledge repository that would ultimately encompass cases spanning various business sectors.

The work evolved in creating a taxonomy used to consider different dimensions: Sector classifications (covering 20 domains from Administration and Finance to Healthcare and Technology), Modularity categories (which included archived-modules, BPMn, entity, field, and standard-module), DSL Types (such as business-rules, CQL, DMN, field-dependency, gendoc, validation-rules), and specialized Area and Application taxonomies. This comprehensive classification structure supplied the organizational framework necessary for efficient retrieval and cross-domain analysis of cases.

With this taxonomy in place, we prepared an associated DSL-based solutions. Each case underwent documentation according to a standardized approach, capturing the problem definition along with its contextual factors, a comprehensive solution description including syntax and implementation specifics, concrete real-world examples demonstrating practical application. This documentation process demanded close collaboration with domain

experts from various sectors to ensure the accuracy, relevance, and completeness of the compiled cases.

As the collection expanded, patterns began to surface across sectors, revealing how similar business challenges presented themselves in diverse domains and how DSL solutions could be adapted and transferred between contexts. These cross-domain insights proved especially valuable, indicating the potential for a more integrated approach to business problem-solving than what had been previously acknowledged in the literature. The significant volume of cases—ultimately surpassing 5,000—posed both challenges and opportunities: challenges regarding organization and accessibility, but opportunities for detecting patterns and relationships that would have otherwise gone unnoticed in smaller collections.



4.3 Key Findings and Implications from Knowledge Base Development

Our thorough evaluation process, which included domain specialists, user testing sessions, and a systematic analysis, generated several vital insights that would profoundly shape the course of our research. The multi-faceted taxonomy we created proved to be an invaluable organizational framework, allowing users to methodically navigate through thousands of cases spanning various sectors, DSL types, and application domains. Nevertheless, as the knowledge base grew to encompass over 5,000 cases, the shortcomings of a static json/text representation became increasingly evident. Users can find challenging to understand and explore intricate cases with numerous

relationships and dependencies, as navigation difficulties impeded their ability to derive meaningful insights from the extensive repository we had curated.

These evaluation insights uncovered a fundamental conflict between content richness and accessibility. Although we had effectively built a comprehensive knowledge base, the static representation model significantly restricted its practical applicability. Domain experts validated the accuracy and relevance of individual cases but pointed out difficulties in grasping cross-case relationships and extracting domain-specific patterns. User testing sessions highlighted particular challenges when transitioning between related cases, with participants often losing context or overlooking essential connections that were implicitly present yet not visually represented.

This serendipitous realization—that the value of information was limited not by content quality but by representational constraints—became the impetus for our future research trajectory. It was clear that a more advanced method for visualizing and interacting with the knowledge base was vital, especially one capable of dynamically illustrating relationships and dependencies among cases. The necessity for visualization was not simply a matter of interface preference but a crucial requirement for unlocking the complete potential of the knowledge we had gathered. This understanding directly influenced our choice to investigate interactive visualization techniques in the subsequent phase of research, ultimately leading to the fortunate discovery of the intensional framework approach that would revolutionize our comprehension of low-code platforms.

5. Phase 2: Interactive UI Implementation

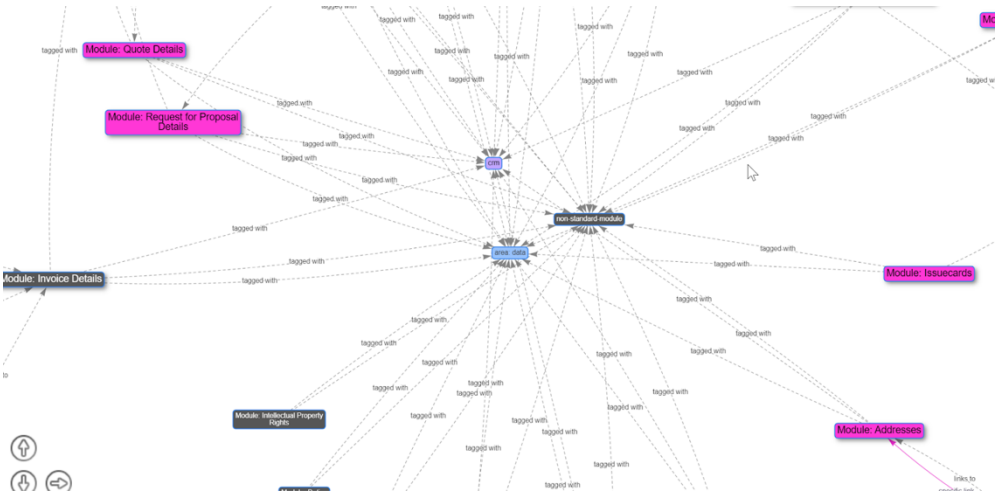
Building on the insights gained from Phase 1, which uncovered shortcomings in the static depiction of the knowledge base, Phase 2 (2018-2020) concentrated on developing an interactive user interface aimed at improving the understanding and exploration of the gathered cases. This phase marked a substantial transition from merely emphasizing content to also considering the manner in which that content is presented and engaged with.

Our shift from a static knowledge base to a dynamic framework was driven by an extensive analysis of user interface needs, which highlighted four essential requirements (visual relationship representation, intuitive navigation, flexible organization, and interactive exploration) balanced with technical demands for

responsiveness across the cases, cross-platform compatibility, extensibility, and maintainability—prompting us to choose TiddlyWiki for its self-contained design, innate flexibility, wiki-like structure of interconnected tiddlers, and JavaScript foundation that enabled the custom extensions necessary to transform our repository into an intensional approach that would redefine our low-code platform development.

5.1 Dynamic Visualization Through TiddlyWiki

TiddlyWiki serves as the technological backbone for our dynamic visualization capabilities, with its open-source, self-contained architecture functioning entirely within browsers, storing all content in a single HTML file, utilizing a non-linear information structure of interconnected "tiddlers" that act as nodes in a network, supporting our multi-dimensional taxonomy through its tagging system, allowing for the representation of complex dependencies through unique transclusion features, facilitating advanced visualizations ranging from network diagrams to hierarchical trees, democratizing access through its portable nature, and ultimately evolving beyond a basic wiki into the cornerstone of our intensional framework where relationships and properties actively generate insights that have fundamentally reshaped our approach to low-code platform development.



5.2 Findings and Transformation to Intensional Framework

Our deployment of the TiddlyWiki-based visualization layer provided deep insights that significantly altered our comprehension of the system that could be constructed. What started as a practical solution to improve accessibility and exploration of our knowledge base unfolded into something far more

profound—an emergent intensional framework with distinctive characteristics that surpassed our initial vision.

The most notable realization was the self-referential, quine-like quality of the system. Similar to a quine in programming—a program that outputs its own source code—our TiddlyWiki implementation could depict and modify its own structure. This attribute allowed the system to evolve naturally through user interactions, generating new connections, insights, and structures that were not explicitly encoded.

These fortuitous revelations were not genuinely accidental but rather arose from the deliberate design of the system's interconnectivity, directly informing the subsequent research phase and enhancing access to knowledge through a discovery mechanism.

6. Phase 3: Intensional Framework Evolution

Building upon the insights acquired during Phase 2, our third research phase (2020-2022) concentrated on purposefully advancing the system from an enriched knowledge base into a fully developed intensional framework. This phase represented a significant paradigm shift in our understanding of the system, evolving it from a mere repository of information into a vibrant, generative framework capable of producing new structures, insights, and solutions through its inherent properties and relationships.

Our shift from serendipity to intentionality constituted a crucial reconceptualization of our system, as we acknowledged that users' seemingly random discoveries were, in fact, emergent characteristics of our framework's interconnected structure. By placing a greater emphasis on intensionality rather than extensionality, we redefined our static compilation of over 4,000 cases into a dynamic framework where relationships provided insights that exceeded explicit encoding, reminiscent of a quine program that generates its own code. We intentionally enhanced four essential attributes: interconnectivity through various types of semantic relationships, bidirectional links, and automated connection discovery; adaptability via interactive learning, collaborative content development, and contextual awareness; scalability through a federated architecture, modular extensions, and progressive rendering; and purpose-driven design with goal-oriented navigation paths and context-sensitive recommendations. The technical implementation expanded TiddlyWiki with bespoke plugins and an advanced relationship engine that constituted the core of our intensional framework.

6.1 Limitations and Challenges

Notwithstanding the substantial advantages exhibited by our intensional framework methodology, several limitations and challenges necessitate careful consideration:

Scalability presents a challenge when addressing exceedingly large knowledge bases comprising millions of relationships. Although our implementation successfully handled over 5,000 cases, further optimization is essential for accommodating significantly larger datasets.

The offline synchronization capabilities of our TiddlyWiki-based implementation showed constraints, particularly when multiple users enacted conflicting modifications to the same content. More advanced synchronization mechanisms will be requisite for extensive collaborative applications.

The learning curve associated with the intensional framework methodology proved to be more pronounced than that of traditional knowledge bases, especially for users familiar with hierarchical or keyword-oriented information retrieval. The incorporation of additional onboarding and guidance features would enhance usability for novice users.

The computational resources demanded for relationship discovery, semantic analysis, and adaptive recommendations escalated considerably as the knowledge base expanded. Optimizing these processes for performance will be critical for broader deployment.

These limitations underscore avenues for future refinement and enhancement of the intensional framework methodology, particularly as it scales to accommodate larger knowledge bases and more heterogeneous user communities.

8. Future Directions: AI Integration

Our research now aims to refine our intensional framework by strategically incorporating advanced AI technologies. This evolution builds on our established foundation while addressing its limitations.

Generative AI technologies present significant opportunities to enhance our framework's capabilities. Implementing context-aware code generation allows us to translate natural language requirements into relevant code segments, utilizing large language models. The framework's contextual understanding supports these generative abilities, while advanced neural networks improve pattern recognition across various business domains, fostering cross-domain insights.

We are also creating conversational interfaces to facilitate natural language interactions, simplifying access to complex relationship networks for users. Importantly, machine learning algorithms can expedite relationship discovery by automatically identifying and recommending connections between entities based on semantic analysis.

Retrieval-Augmented Generation (RAG) approaches provide promising enhancement opportunities. RAG's architecture, which combines generative capabilities with contextual retrieval, aligns seamlessly with our framework's design principles. By retrieving pertinent cases and relationships, RAG generates more contextually appropriate outputs and allows for the integration of new knowledge based on user contributions and automated discovery.

RAG also enhances explainability for business users by providing specific cases and relationships that support its recommendations, ensuring transparency in decision-making. Its hybrid nature balances innovation with reliability, maintaining adherence to established patterns while enabling novel applications.

Our research roadmap includes several interconnected initiatives: developing proof-of-concept implementations for specific use cases, creating robust evaluation frameworks, addressing ethical considerations, optimizing for specific business domains, and exploring collaborative intelligence models that combine human expertise with AI. Through this integrated AI enhancement approach, it’s possible to establish a more intelligent, adaptive, and supportive foundation for low-code development, preserving the relationship-driven insights of our intensional framework while significantly broadening its generative capabilities.

9. Conclusion

Our extensive research undertaking over multiple years has meticulously charted the transition from a conventional knowledge base to an intentional

framework designed for low-code platforms. This transformation signifies not only a technological progression but also a profound reconceptualization of the organization, presentation, and application of domain knowledge within software development environments.

The intentional framework methodology, prioritizing relationships and attributes rather than isolated content elements, provides a robust model for navigating the complexities inherent in contemporary software development. By empowering users to identify patterns, connections, and applications that traverse traditional domain boundaries, this methodology promotes more innovative and efficient development methodologies.

Our practical application utilizing TiddlyWiki illustrates how comparatively straightforward technologies, when strategically organized, can construct intricate knowledge frameworks that surpass the sum of their individual components. The fortuitous identification of intentional properties arising from our preliminary visualization efforts underscores the potential for unforeseen insights when conventional methodologies are augmented with novel perspectives and technologies.

The incorporation of Generative AI and Retrieval-Augmented Generation is poised to further augment the functionalities of our intentional framework, establishing an even more formidable foundation for low-code development. These innovations are expected to foster more intelligent, contextual, and creative software development processes, thereby further democratizing access to advanced application development capabilities.

In summary, the transition from a knowledge base to an intentional framework exemplifies a broader paradigm applicable to numerous information systems: the potential for well-organized repositories to transform into dynamic, generative frameworks when enhanced with suitable visualization, relationship modeling, and interaction capabilities. This paradigm presents a promising trajectory for the future of knowledge management and software development, indicating systems that not only retain information but also actively engage in the generation of new insights and solutions.