# WHITE PAPER

## AI-Driven Legacy Application Modernization

*How Generative AI Transforms Code Migration from Legacy Systems to Modern Architectures*

*Prepared by:* **BotMan AI**

*Product:* **Codacle — Application Modernization Platform**

*Date:* March 2026

*Version:* 1.1

## 1. Executive Summary

Organizations worldwide are trapped in a costly paradox. The legacy applications that power their most critical business processes—built decades ago in technologies like COBOL, Delphi, Oracle Forms, and Visual Basic—are becoming increasingly expensive to maintain, difficult to secure, and impossible to evolve. Industry research consistently shows that enterprises allocate 60 to 80 percent of their IT budgets simply to keeping these aging systems running, leaving a fraction for innovation and growth.

The legacy application modernization market reflects this urgency. Valued at approximately USD 25 billion in 2025, it is projected to grow at a compound annual growth rate exceeding 15 percent through the end of the decade. Organizations across financial services, healthcare, retail, manufacturing, and government are all racing to replace monolithic, outdated architectures with modern, cloud-native systems that can support microservices, API-first design, and real-time data processing.

Yet traditional modernization approaches—manual rewrites conducted by large consulting teams over multi-year timelines—remain slow, expensive, and prone to failure. Critical business logic is lost in translation, timelines stretch beyond initial estimates, and the resulting code often fails to match the quality standards of the original system.

Codacle, developed by BotMan AI, offers a fundamentally different approach. By leveraging Generative AI across the entire modernization lifecycle—from legacy code analysis and knowledge graph construction to architecture design and automated code generation—Codacle reduces modernization time and cost by 50 to 90 percent while producing clean, maintainable, well-documented code in modern technology stacks.

This white paper examines the legacy application challenge, introduces the Codacle methodology, presents real-world results from enterprise-scale migrations, and explains how AI-driven modernization is reshaping what organizations can expect from their transformation initiatives.

## 2. The Legacy Application Challenge

### 2.1 The Scale of the Problem

Legacy systems are far more pervasive than many executives realize. Applications built in COBOL, Delphi, Visual Basic, Oracle Forms, and similar technologies continue to process transactions, manage supply chains, and serve customers across virtually every industry. These systems were often well-engineered for their era, but the technology landscape has shifted dramatically since they were deployed.

The financial burden is staggering. Research from multiple industry sources indicates that large enterprises routinely spend 60 to 80 percent of their total IT budgets on maintaining and operating legacy systems. The U.S. federal government, for example, has historically allocated roughly 80 percent of its IT spending to operations and maintenance of aging infrastructure, leaving only 20 percent for development and modernization. In the private sector, technical debt alone is estimated to consume 40 percent of IT budgets, and this figure compounds at approximately 20 percent per year when left unaddressed.

The talent crisis compounds the financial pressure. The pool of developers skilled in legacy languages is shrinking rapidly as experienced professionals retire. Universities have largely dropped COBOL and similar languages from their curricula. Organizations find themselves competing for an ever-smaller talent pool, paying premium salaries for expertise that is becoming rarer each year.

## 2.2 Risks and Costs of Inaction

The consequences of delaying modernization extend well beyond budget overruns. Organizations clinging to legacy systems face three interconnected categories of risk:

- **Maintenance and operational risk.** Legacy applications frequently lack adequate documentation. The developers who originally built them have often left the organization, taking critical institutional knowledge with them. Every modification carries the risk of unintended side effects in tightly coupled, monolithic codebases. Routine maintenance tasks that would take hours in a modern system can consume days or weeks.
- **Inflexibility and competitive risk.** Monolithic architectures cannot adapt quickly to new business requirements. Integrating with modern APIs, supporting mobile interfaces, or deploying to cloud environments requires extensive workarounds. Competitors operating on modern stacks can release new features in days while legacy-dependent organizations struggle with release cycles measured in months.
- **Security and compliance risk.** Legacy systems are prime targets for cyberattacks because they often run on unsupported software that no longer receives security patches. Outdated authentication mechanisms, unencrypted data flows, and rigid architectures make it difficult to comply with evolving regulations such as GDPR, HIPAA, PCI DSS, and SOX. Financial institutions, for instance, allocate a disproportionate share of their IT budgets to compliance maintenance alone.

## 2.3 Why Traditional Migration Falls Short

Organizations that do commit to modernization often discover that the traditional approach—assembling large teams of developers to manually rewrite applications—introduces its own set of problems.

Manual rewrites are slow. A typical enterprise migration project spans two to five years, during which the legacy system must continue operating and the business must manage parallel environments. They are expensive, frequently exceeding initial budget estimates as hidden complexity emerges. Most critically, they are risky: developers working from incomplete documentation and fragmented knowledge of the original system routinely miss edge cases, misinterpret business rules, or introduce inconsistencies that only surface in production.

Code assistant tools—the current generation of AI-powered development aids—help individual developers write code faster, but they do not solve the fundamental challenge. They lack the ability to understand the complete context of a legacy application: the relationships between hundreds of procedures, functions, tables, and business rules that together define how the system operates. Without this holistic understanding, modernization remains a piecemeal, error-prone process.

What organizations need is an approach that combines the deep contextual understanding of the entire legacy codebase with the speed and consistency of automation. This is precisely the gap that Codacle was designed to fill.

## 3. The Codacle Approach: AI-Driven Modernization

Codacle is BotMan AI's proprietary platform for accelerating legacy application modernization through Generative AI. Unlike conventional tools that assist developers with isolated coding tasks, Codacle operates at the application level—analyzing, understanding, and transforming entire systems from their original technology stack to modern architectures.

The core principle behind Codacle is straightforward: map in detail every step that a skilled human team would perform if conducting the migration manually, then deploy AI-based accelerators that reproduce each step autonomously while preserving the ability for human experts to validate and adjust at every stage. This is implemented through a proprietary pipeline of over 60 deterministic steps, each powered by Generative AI with precisely the right context from the knowledge graph—no more, no less.

### 3.1 Knowledge Graph Construction

The foundation of Codacle's approach is a comprehensive knowledge graph that captures the complete structure, behavior, and relationships within the legacy application.

During the analysis phase, Codacle decomposes the source code and database into their constituent elements—packages, classes, methods, functions, stored procedures, tables, views, and triggers—according to the conventions of the original programming language. Each element receives detailed documentation in both natural language and pseudocode, generated automatically by the AI from the source code itself.

Critically, the knowledge graph also maps all relationships between elements: which methods belong to which classes, which methods call other methods, which procedures read from or write to which tables, how data flows between components, and how user interactions trigger chains of operations. This relational mapping is what distinguishes Codacle from simpler code analysis tools. It provides the AI with a structured, semantically rich representation of the entire application —not just individual files or functions, but the complete web of dependencies and interactions that define the system's behavior.

> ***Knowledge graphs provide a structured way of understanding relationships between entities, which leads to improved contextual understanding and significantly reduces the risk of AI hallucinations. By having all connections between elements mapped, no relevant information is left out during the transformation process.***

### 3.2 Target Architecture Design

With the knowledge graph in place, Codacle moves to architecture design. Rather than applying a one-size-fits-all template, the platform analyzes the graph to identify the most relevant business modules and generates a new microservices architecture centered around those modules.

The target architecture features a clean separation between frontend and backend, with RESTful APIs defining the contract between layers. Codacle supports a wide range of modern technology stacks to accommodate each organization's preferences and existing infrastructure:

- **Frontend technologies:** React, Angular, and other modern JavaScript frameworks.
- **Backend technologies:** Java (Spring Boot), .NET 8, Node.js, and other enterprise-grade platforms. Other stacks available upon request.

This architecture design phase includes a deliberate review checkpoint. Before any code is generated, human experts—from BotMan AI and optionally from the client organization—review the proposed architecture to ensure it aligns with the client's design principles, infrastructure constraints, and long-term technology strategy.

## 3.3 Automated Code Generation

Code generation is where Codacle's knowledge graph delivers its greatest advantage. Because the graph connects each database table to all the original application code that reads from or writes to it, the AI can understand and replicate the original business logic within the new architectural structure.

Backend generation. Codacle automatically generates the complete backend layer in the target language and framework. In a Spring Boot migration, for example, this includes controllers that expose the RESTful API endpoints, services that implement business logic, entities that model the data domain, and repositories that handle data access. Each generated component encapsulates the business rules that were previously embedded in legacy procedures and functions, but now organized according to modern software engineering practices.

Frontend generation. Codacle uses specialized Generative AI agents to analyze the design, layout, and behavior of the original application's screens, then generates equivalent user interfaces in the target frontend framework. The AI separates the original code for each user interaction into front-end behaviors (which are replicated at the UI level) and back-end behaviors (which are replaced with calls to the newly created APIs). The result is a modern, responsive interface that preserves the functional flow users expect while taking advantage of contemporary UI capabilities.

## 3.4 Human-in-the-Loop Validation

Codacle is not a black-box automation tool. At every stage of the process, human experts can review, adjust, and validate the AI-generated output:

- **Architecture review:** Before code generation begins, the proposed microservices architecture and API design are reviewed against client standards and best practices.
- **Code quality review:** Generated code is inspected for correctness, readability, and adherence to coding standards. Adjustments can be made by both BotMan AI engineers and client developers.
- **Assisted testing:** Codacle leverages Generative AI tools to facilitate and accelerate the development of test cases and automate their execution against the migrated solution, ensuring functional parity with the original system.

- **Automated validation:** An autonomous validation pipeline reviews hundreds of data points from the original source code to validate accuracy in the migrated code, leaving evidence of each validation.

This human-in-the-loop model ensures that the speed advantages of AI automation do not come at the expense of quality or alignment with business requirements.

### 3.5 Barbara: Intelligent Code Assistant

As part of the Codacle platform, BotMan AI offers Barbara—an AI-powered agent that traverses the knowledge graph in real time to answer complex queries about the codebase. Development teams can query Barbara about information flows between components, relationships between objects, unused or redundant elements, and which components would need to be modified to implement a new feature.

Barbara integrates directly into the developer's IDE as an MCP tool, supporting Visual Studio Code, Cursor, Windsurf, and other modern development environments. It serves as a living documentation layer that makes the knowledge graph accessible and actionable for development teams—both during the migration project and long after it is complete.

## 4. Use Cases and Capabilities

Codacle's AI-driven approach applies across a broad range of modernization scenarios. While full application migration is the most common engagement, the platform's knowledge graph and code generation capabilities extend to several adjacent use cases.

| Use Case | Description |
|---|---|
| **Full Application Migration** | End-to-end migration of legacy applications from technologies such as Delphi, Visual Basic, COBOL, and Oracle Forms to modern architectures including Spring Boot, .NET 8, Node.js, React, and Angular. |
| **Technical and Functional Documentation** | Automated generation of comprehensive documentation from source code, covering application architecture, business logic, data flows, and integration points. |
| **Intelligent Data Catalog (Metron)** | Through the Metron module, automated creation of enriched data dictionaries from database schemas, sample data, query logs, and optionally application source code, with PII detection and compliance readiness. |

### 4.1 Metron: Accelerating Data Governance

Metron is a specialized module within the Codacle ecosystem designed to accelerate data governance initiatives. It automatically generates enriched data dictionaries by combining multiple sources of knowledge: database schemas (DDL), sample data, SQL query logs, stored procedures, and optionally the application knowledge graph itself.

For each field in the database, Metron produces a description, classifies the sensitivity of the information (including detection of direct and indirect PII), and provides evidence and a degree of certainty for each

generated description. When the application knowledge graph is available, accuracy can reach 90 to 95 percent, because the AI understands not just the database structure but also how the application reads, writes, and transforms the data.

Metron outputs are designed for integration with governance platforms such as Collibra, Alation, Informatica EDC, Atlan, and dbt Docs, with export options including YAML, CSV, and API. The module transforms what traditionally takes weeks of manual effort by data stewards into a process that completes in hours, producing living documentation that can be updated automatically as the database schema evolves.

## 5. Case Study: Enterprise-Scale Delphi Migration

### 5.1 Client Profile

Forus is a multinational retailer in fashion, operating across Chile, Peru, Colombia, and Uruguay, with more than 400 physical stores. Like many established retailers, Forus had built a substantial technology estate over decades, with core business applications developed in Delphi.

### 5.2 The Challenge

Forus needed to modernize 500 Delphi forms and 70 batch processes—comprising 550,000 lines of code accumulated over 32 years—migrating them to a modern architecture based on Spring Boot for the backend and React for the frontend. The scale of the project—encompassing the full range of retail operations from inventory management to e-commerce integration—made a traditional manual rewrite impractical within acceptable timelines and budgets. A manual rewrite was projected to take 16 years.

### 5.3 The Codacle Approach

BotMan AI deployed Codacle to execute the migration through its four-phase methodology:

1. **Knowledge graph construction.** Codacle analyzed the entire Delphi codebase and supporting databases, building a comprehensive graph of all application elements—procedures, functions, forms, tables, stored procedures—and their relationships. Each element was documented in natural language and pseudocode.

2. **Architecture design.** Using the knowledge graph, Codacle identified the core business modules and designed a new microservices architecture with RESTful APIs. The architecture was reviewed and approved by the Forus technical team before proceeding to code generation.

3. **Code generation.** Codacle generated the Spring Boot backend—controllers, services, entities, and repositories—replicating the business logic from the original Delphi procedures within the new API structure. Frontend agents then analyzed the original Delphi forms and generated equivalent React interfaces, connected to the new backend services.

4. **Validation and testing.** Human experts reviewed the generated code at each stage. Generative AI tools were used to develop test cases and automate their execution, ensuring functional parity between the original and modernized applications.

### 5.4 Results

The Codacle-driven migration delivered measurable outcomes across every dimension that matters to enterprise stakeholders:

- **Time and cost reduction:** The migration was completed in 14 months with 50 to 90 percent less time and cost compared to what a traditional manual rewrite would have required—a project that had been projected to take 16 years.
- **Code quality:** The generated code was clean, readable, and maintainable, following modern development best practices and accompanied by comprehensive documentation.
- **Modern architecture:** Forus gained a microservices-based system that supports independent scaling of components, straightforward integration with new technologies, and rapid feature development.
- **Reduced risk:** The combination of AI-driven automation, structured validation checkpoints, and compressed timelines significantly reduced the risks typically associated with large-scale migration projects.
- **Automatic documentation:** Codacle generated detailed documentation for the new codebase, reducing the knowledge transfer burden and making the system immediately accessible to development teams.

**50–90%**

**Reduction in migration time and cost compared to traditional manual rewrite**

## 6. Security and Data Protection

Organizations considering AI-driven modernization rightfully want to understand how their source code and data will be handled throughout the process. BotMan AI has designed its security practices to meet the expectations of enterprise clients across regulated industries.

### 6.1 Code Storage and Handling

All processes are executed on secure networks and computers. Client source code is stored exclusively on these secured systems and in Microsoft OneDrive folders for the duration of the project. Upon project completion, all copies are permanently deleted from BotMan AI storage. At no point is source code shared with unauthorized parties or retained beyond the agreed project timeline.

### 6.2 AI and LLM Usage

Codacle's use of large language models is restricted to enterprise-grade APIs provided by Azure, AWS, Google Cloud, and Anthropic. These providers offer enterprise security commitments including data encryption in transit and at rest, no use of customer data for model training, and compliance certifications relevant to regulated industries. If a client requires the exclusion of any specific provider, BotMan AI can accommodate this constraint.

BotMan AI does not currently use local or open-source LLMs for modernization work. Internal testing has shown that enterprise API-based models deliver substantially higher quality for the complex reasoning and code generation tasks that modernization demands.

### 6.3 Authentication and Authorization

For migrated applications, BotMan AI's default approach is to maintain the same authentication and authorization method used by the legacy application, ensuring continuity for end users and minimizing disruption. Alternatively, the team can integrate with modern authentication standards such as Active Directory, or build entirely new authentication mechanisms tailored to the client's security requirements.

### 6.4 Data Access

Code migrations do not require access to production database data. BotMan AI needs access only to test or synthetic data during the testing phase of the migration. This separation ensures that sensitive production data remains within the client's security perimeter throughout the engagement.

## 7. Conclusion and Next Steps

Legacy application modernization is no longer optional for organizations that want to remain competitive, secure, and compliant. The costs of maintaining aging systems—measured in budget consumption, security exposure, talent scarcity, and lost agility—compound with every year of delay.

Codacle transforms what has traditionally been a multi-year, high-risk, high-cost endeavor into a structured, AI-accelerated process that delivers results in a fraction of the time. By building a complete knowledge graph of the legacy application, designing a modern target architecture, and generating clean code across both frontend and backend layers—all with human expert oversight at every stage—Codacle addresses the root causes of modernization failure: incomplete understanding, slow execution, and inconsistent quality.

The results speak for themselves: 50 to 90 percent reductions in time and cost, enterprise-grade code quality, comprehensive documentation, and modern architectures that position organizations for years of future innovation.

### Getting Started

BotMan AI offers a structured path from initial assessment to full-scale modernization:

- **Discovery session.** A collaborative session to assess your legacy portfolio, identify priority applications, define target architectures, and estimate scope and timelines.
- **Proof of concept.** A focused pilot migration of a representative application or module, demonstrating the Codacle methodology and producing measurable results within weeks.
- **Full-scale engagement.** A structured modernization program covering your complete application portfolio, with clear milestones, quality gates, and ongoing collaboration between BotMan AI and your technical team.

**Ready to modernize your legacy applications?**

**Contact BotMan AI to schedule a discovery session and see Codacle in action.**