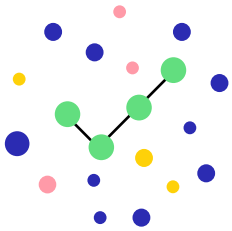
Progress® OpenEdge®

Progress OpenEdge AI Assistant

WHITEPAPER

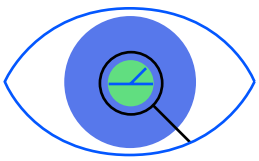


A New Era of Agile, AI Driven Development

For some developers, working with AI is already a straightforward, daily interaction but for many others, it's still unknown territory. In the Progress OpenEdge ecosystem, where long-established codebases, evolving business requirements and the ongoing adoption of new technologies converge, we also find ourselves at a time of digital transformation in enterprise applications.

The Progress OpenEdge AI Assistant emerges as a direct response to new challenges where AI becomes a meaningful part of daily workflows. We created a robust, agile framework that harnesses the power of Generative AI (GenAI) and Retrieval-Augmented Generation (RAG) to elevate the productivity and agility of OpenEdge developers, while supporting critical modernization and upgrade initiatives.

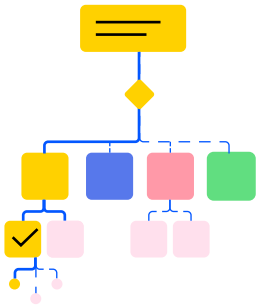
This whitepaper explores the journey behind the Progress OpenEdge AI Assistant, the lessons that shaped its design and the transformative impact it brings to OpenEdge development, especially in the context of OpenEdge 12 upgrades and modernization.



The Vision: Bridging AI and OpenEdge Development

The Progress OpenEdge AI Assistant was conceived to address a fundamental gap: while GenAI and large language models (LLMs) have revolutionized software development, their effectiveness is limited in specialized domains like OpenEdge Advanced Business Language (ABL), where public models lack deep, domain-specific knowledge. Recognizing this, the team behind the toolkit, brought together for their cross-functional expertise, began to scope out the problem and potential solution(s).

The goal was clear: to deliver a tool that could automate repetitive tasks, improve code quality, accelerate modernization and provide actionable guidance, ultimately making OpenEdge developers more productive and agile in their daily work.



The Development Journey: Lessons That Shaped the Framework

Early experiments revealed that generic LLMs, even when fine-tuned, struggled with the nuances of ABL and the specific workflows of OpenEdge development. Training a custom LLM was quickly deemed impractical, given the scarcity of large, high-quality ABL datasets and the prohibitive costs involved.

This shortfall arises primarily because generic LLMs, such as GPT-4 or DeepSeek-Coder, are trained on vast public code repositories like GitHub or Stack Overflow, where OpenEdge ABL is scarcely represented. As a result, these models possess only a superficial or generalized understanding of ABL, leading to frequent inaccuracies when tasked with generating or refactoring ABL code. For instance, generic LLMs often produce syntactically incorrect ABL code or fail to apply new conventions such as the VAR syntax and even more advanced, domain-tuned models require significant augmentation through Retrieval-Augmented Generation (RAG) to achieve acceptable results.

The challenge is compounded by the scarcity of proprietary, high-quality ABL datasets necessary for training or fine-tuning a more specialized model. Within Progress and the broader OpenEdge community, large, curated repositories of ABL code are not readily available. Much of the existing ABL code is proprietary, fragmented across organizations or tied up in legacy systems, making it inaccessible for large-scale training. Even if access were possible, preparing this data for effective model training would be a formidable undertaking. The process would require semantic chunking, breaking code into meaningful units such as DEFINE or PROCEDURE statements and meticulous annotation to embed ABL-specific rules and organizational guidelines. This is a labor-intensive process that demands significant time and expertise. Moreover, the sheer scale of data required to train a competitive LLM is daunting; modern models are trained on petabyte-scale datasets, a volume of data far beyond what is available in the OpenEdge domain.

The financial and operational costs associated with training a custom LLM further underscore the impracticality of this approach for a proprietary language like ABL. Training a state-of-the-art language model from scratch requires thousands of hours of GPU or TPU compute time, with costs easily reaching millions of dollars. For example, training GPT-3 reportedly cost around $5 million, while GPT-4's training budget soared to $100 million or more. Even fine-tuning existing models can be expensive, especially when factoring in the need for specialized machine learning engineers who understand both AI and the intricacies of ABL, a rare and costly skill set. Additionally, because ABL is highly specialized and tailored to specific business applications, the return on such an investment is uncertain, as the improvements in model performance may not justify the substantial outlay.

This realization led to a pivotal shift. Instead of relying solely on generative models, the team adopted a RAG approach. RAG combines the strengths of information retrieval with generative AI, enabling the toolkit to fetch relevant code samples, documentation and organizational guidelines at query time and incorporate them into its responses. This not only improved the accuracy and relevance of the outputs but also enabled the toolkit to adapt to evolving codebases and business rules.

Another key lesson was the importance of data preparation and chunking. Effective RAG requires proprietary data to be semantically chunked and embedded, allowing for fast, context-aware retrieval. The team experimented with chunking strategies based on ABL statements and procedures, learning that the quality and structure of the dataset directly impacted the toolkit's performance.

Equally important was the understanding that developer experience matters. Early prototypes that relied on chat interfaces were insufficient; developers needed AI-powered assistance embedded directly in their IDEs, such as Visual Studio Code, to enable seamless, in-context support. This led to a focus on deep IDE integration, workflow automation and extensibility, making the toolkit a natural extension of the developer's environment.

Finally, the journey underscored the value of workflow-driven design. By embedding organizational rules, quality checks and step-by-step workflows (such as creating new ABL classes or refactoring older code), the toolkit could deliver not just syntactically correct code but maintainable, secure and compliant solutions aligned with business objectives.

How the Progress OpenEdge AI Assistant Elevates Productivity and Agility

The Progress OpenEdge AI Assistant transforms the developer experience in several fundamental ways:

1

Automating Routine and Repetitive Tasks

The connector excels at code completion, boilerplate generation, documentation and even automated code reviews. Developers can use natural language prompts to generate ABL code snippets, refactor legacy code to modern syntax or request the creation of temp-tables and enumerations, dramatically reducing manual effort and minimizing errors.

2

Improving Code Quality and Consistency

By leveraging RAG, the connector retrieves context-specific information from proprietary documentation, codebases and guidelines, so that code suggestions are accurate, relevant and aligned with organizational standards. Automated validation and enforcement of rules further support consistency and compliance across projects.

3

Enhancing Knowledge Sharing and Onboarding

The connector serves as a living knowledge repository, offering standardized code snippets, best practices and instant feedback. his accelerates onboarding for new develoerps and boosts confidence and productivity across teams, regardless of experience level.

4

Enabling Real-Time Agility

With deep IDE integration and workflow automation, the connector provides real-time, context-rich support without disrupting productivity. Developers can respond rapidly to changing requirements, leveraging actionable guidance and code suggestions grounded in the latest organizational knowledge.

5

Facilitating Modernization and Testing

A critical function of the connector is its support for modernization. It can refactor older code to contemporary ABL patterns, generate unit tests and assist with security scans and regulatory compliance, streamlining upgrades to OpenEdge 12 and reducing technical debt.

Supporting OpenEdge 12 Upgrade and Modernization

The Progress OpenEdge AI Assistant is a powerful catalyst for organizations embarking on the journey to OpenEdge 12. Upgrading to this latest version brings significant benefits—enhanced security, improved performance, modern DevOps capabilities and support for web-based architectures—but also introduces challenges, especially for teams managing extensive and long-standing codebases.

The connector addresses these challenges head-on:

Modernization of Historic Code

It can refactor mature ABL code to adopt new OpenEdge 12 syntax and best practices, such as converting variable declarations or updating temp-table definitions. This automation reduces manual effort, creates consistency and accelerates the migration process.

Accelerated Migration and Testing

The connector scans code for deprecated constructs, flags incompatibilities and suggests changes needed for OpenEdge 12 compatibility. It guides developers through structured workflows for creating new classes, updating APIs or restructuring code for PAS for OpenEdge, minimizing migration risks and post-upgrade rework.

Modern DevOps and CI/CD Integration

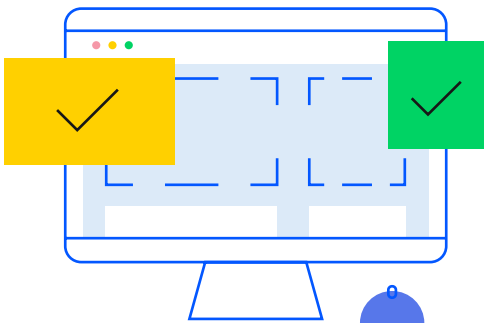
OpenEdge 12 introduces robust DevOps and CI/CD tools. The connector provides code snippets, configuration templates and integration guidance, enabling teams to adopt modern build and deployment practices with confidence.

API and Interoperability Modernization

With OpenEdge 12 offering support for OpenAPI and REST, the connector helps developers expose business logic as RESTful services and update code to leverage new interoperability features, so that applications are future-ready.

Reducing Risk and Cost

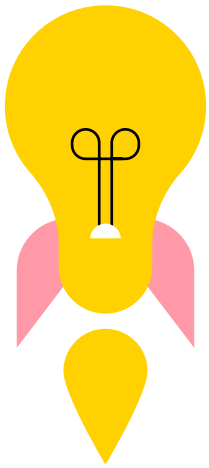
By automating error-prone migration steps and acting as an always-available knowledge base, the connector reduces the risk of human error, accelerates timelines and lowers the overall cost of modernization.



A Connector Built for the Future

The Progress OpenEdge AI Assistant is more than a productivity tool—it is a strategic enabler for organizations seeking to modernize, innovate and compete in a rapidly evolving digital landscape. The system's layered, extensible architecture is engineered to accommodate not only OpenEdge but also with other Progress frameworks and languages considered, optimizing reuse and safeguarding investments for the future.

The journey behind the Connectors development furthered a culture of continuous learning, real-world testing and risk-aware design. By focusing on data readiness, workflow automation and deep integration, the tool is poised to support the evolving needs of developers and organizations for years to come.



What's Next in AI for OpenEdge

The Progress OpenEdge AI Assistant isn't just a framework; it's the beginning of a continuously evolving journey in AI-driven development. By blending Generative AI with coding best practices, frameworks and human expertise and grounding it all in workflow automation, we've created more than a productivity booster; we've built a dynamic framework designed to adapt and grow alongside your business.

As models improve and AI capabilities advance, so too will the Connector. This is not a one-time solution but a living, iterative platform that evolves to meet your changing needs, so that every developer, whether modernizing legacy code or pushing into new frontiers, has the confidence, speed and context to deliver their best work.

As organizations embrace the opportunities of Generative AI, OpenEdge 12 and beyond, the Connector stands as a trusted partner, automating the mundane, empowering the innovative and enabling every developer to deliver their best work, faster and with greater confidence.

Contact us to learn more about the Progress OpenEdge AI Assistant

About Progress Software

Progress Software (Nasdaq: PRGS) empowers organizations to achieve transformational success in the face of disruptive change. Our software enables our customers to develop, deploy and manage responsible AI-powered applications and personalized digital experiences with agility and ease. Businesses of all sizes get a trusted provider in Progress, with the products, expertise and vision they need to turn AI disruption into a competitive advantage. Millions of developers and technologists at hundreds of thousands of organizations depend on Progress every day. Learn more at www.progress.com

Worldwide Headquarters

Progress Software Corporation
15 Wayside Rd, Suite 400, Burlington, MA 01803, USA
Tel: +1-800-477-6473

facebook.com/progresssw
twitter.com/progresssw
youtube.com/progresssw
linkedin.com/company/progress-software
progress_sw_

Progress