AI CODE GENERATION TOOLS — VALUE DRIVERS

Calculating Value From AI Code Generation Utilities

LIMRA LOMA

Navigate With Confidence

Contents

1.0 Introduction

1.1 AI Code Generation Tools (AICGT)

As artificial intelligence (AI) becomes increasingly more sophisticated, its impact on the workforce across the insurance value chain — including within information technology (IT) — is inevitable. AI-driven decision-making and AI products that augment jobs can significantly change job functions, including from the use of AI-based code generation tools within IT. AICGTs are AI-based utilities that can author, complete, validate, and rewrite software code. IT developers can request specific features and functionality, and these utilities can automatically generate code for the requested functionality. Allowing integration with a firm's existing software development infrastructure, programming languages, and practices, developers provide AICGTs a description of what they would like the code to do, and AICGTs generate code that is congruent to a company's software development guidelines.

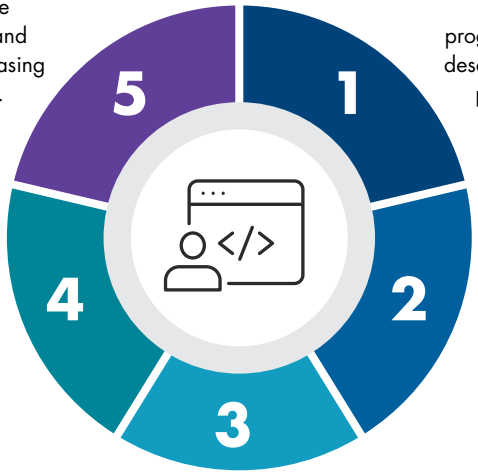
Just like with the use of generative AI (GenAI), even though the end output of AICGTs might not be 100 percent perfect, they do provide a solid place to start and expedite the software development lifecycle. Also like GenAI, these utilities are not intended to replace software developers, but to augment them, allowing them to delegate rote, repeatable, operational code-authoring tasks to AICGTs and focus on higher value-add software development activities. Designed to enhance productivity of software developers, improve code quality, optimize performance, and detect and remediate software cybersecurity vulnerabilities, amongst other benefits, AICGTs are becoming increasingly popular within our industry.

Kartik Sakthivel, Ph.D., MS-IT/MS-CS, MBA, PGC-IQ
Vice President & Chief Information Officer, LIMRA and LOMA
ksakthivel@limra.com

The figure below depicts the five main functions that AICGTs allow firms to do with respect to code. Carriers have been selecting AICGTs based on the features and capabilities each tool offers. Note that not every commercially available AICGT offers every feature. For instance, some AICGTs are primarily code generation utilities and do not support code conversion from one programming language to another. Others provide a holistic approach to software development and deployment, offering seamless interoperability with existing Continuous Integration/Continuous Deployment (CI/CD) pipelines (see business explanation in Appendix A). Pricing continues to be a key consideration for selection of an AICGT. A core requirement for selection of a AICGT is its ability to seamlessly integrate with your existing IT development tools and infrastructure. AICGTs are used in the development of both internally and externally facing software applications. These applications would likely contain Personally Identifiable Information (PII) or Sensitive Personally Identifiable Information (SPII) data. PII is any information that can be used to identify an individual, either on its own or when combined with other data such as full name, email address, phone number, home address, etc. SPII is a subset of PII that, if lost, compromised, or disclosed without authorization, could result in substantial harm, embarrassment, inconvenience, or unfairness to an individual, such as Social Security number, driver's license number, passport number, financial information, medical records, etc. As such, information security and cybersecurity considerations such as access control, encryption, etc., are of great importance.

REVIEW
Generates code that remedies issues such as defects, security vulnerabilities, performance bottlenecks, etc. by automatically reviewing and improving code, thereby saving time and increasing quality, performance, and security of code.

GENERATION
Generate code that you can choose to use as is, with modifications, or discard, in various programming languages using natural language descriptions, and in compliance with established programming patterns and best practices.



LANGUAGE CONVERSION
Allows for conversion of code from one programming language into others, expediting translations and reducing complexity by generating optimized code in the target language.

COMPLETION
Suggests code snippets as developers type by predicting the next most likely code snippet. These predictions are context-aware and require integration with existing code repositories.

MODERNIZATION
Can allow firms to modernize code from legacy systems to our target programming language. This can expedite modernization of legacy systems.

1.2 OVERVIEW

GOAL:

To identify industry-standard levers to allow organizations to leverage these measures into their CBA/ROI templates so they can measure value from AI code generation tools.

The widespread promise of AICGTs to deliver business value has propelled their popularity within our industry. However, deployment of AICGTs without fully understanding their impact can lead to inconsistent usage, underutilization, or wasted investment. AICGTs, if not deployed appropriately and equitably, can also create a two-tiered IT development team within the same firm — where developers who *have* access to these tools outperform their counterparts who might *not* have access to AICGTs, or simply haven't been trained on how to capitalize on them. AICGTs are adding direct business value. Identifying value drivers of AICGTs is important to demonstrate to the C-suite and IT's business partners why IT groups are investing in AI code-generation tools.

A robust assessment of AICGTs value drivers is essential for:

DEMONSTRATING ROI: Being able to justify continued investment by quantifying gains in productivity, cost savings, software quality, time-to-market, cybersecurity, etc.

STRATEGIC BUSINESS ALIGNMENT: Ensuring that AICGTs contribute to broader business and IT goals.

SCALING OPTIMIZATION: Identifying use cases where AI code generation is most effective and scaling it accordingly.

RISK MANAGEMENT: Addressing concerns such as code security, bias, and regulatory compliance in AI generated code.

DEVELOPER EXPERIENCE: Understanding how these tools reduce cognitive load and improve developer satisfaction, recruitment, and retention.

My definition of success with code-generation tools is the answer to the question,

"How aligned is the code being generated from these products meeting business requirements with a high degree of scalability, extensibility, reuse, reliability, cybersecurity, and quality?"

1.3 Common AICGTs

As of Q1 2025, most carriers represented within the AI Governance Group (AIGG), are using GitHub Copilot for AI code generation. Some are leveraging Databricks and Microsoft Copilot. While the following should in no way constitute an endorsement of any or all of these AICGTs, some common AICGTs in use across the industry include GitHub Copilot, Amazon Q Developer (formerly CodeWhisperer) — optimized for Amazon Web Services (AWS) development, Google Gemini, Google Vertex, Google Code Assist, TabNine, Meta's Code Llama (open-source model), etc.

GitHub Copilot is the AICGT that comes up the most in any AICGT conversation within the industry. GitHub Copilot was developed by GitHub in collaboration with OpenAI, and leverages a language model that was trained on diverse code repositories, known as OpenAI Codex. Whereas OpenAI's most popular product, ChatGPT, can be leveraged for code generation, it is not *designed* for code generation and does not integrate into a firm's existing software development infrastructure and tooling. GitHub Copilot's popularity can be attributed to the fact that several firms have been leveraging GitHub, and its most prominent feature is its ability to integrate into most carriers' existing programming/software development infrastructure, including Integrated Development Environments (IDEs) such as Microsoft Visual Studio.

2.0 AICGT Value Drivers

AICGTs are transforming software development across our industry by automating repetitive coding tasks, improving developer productivity, and reducing time-to-market. For carriers to justify investment in these tools and maximize their ROI, they must have a structured approach to evaluating their impact. This inventory is intended to help firms systematically assess, measure, and optimize the value of AICGTs by identifying key value drivers, establishing quantitative and qualitative metrics, and ensuring alignment with strategic business objectives.

2.1 BASIC VALUE DRIVERS

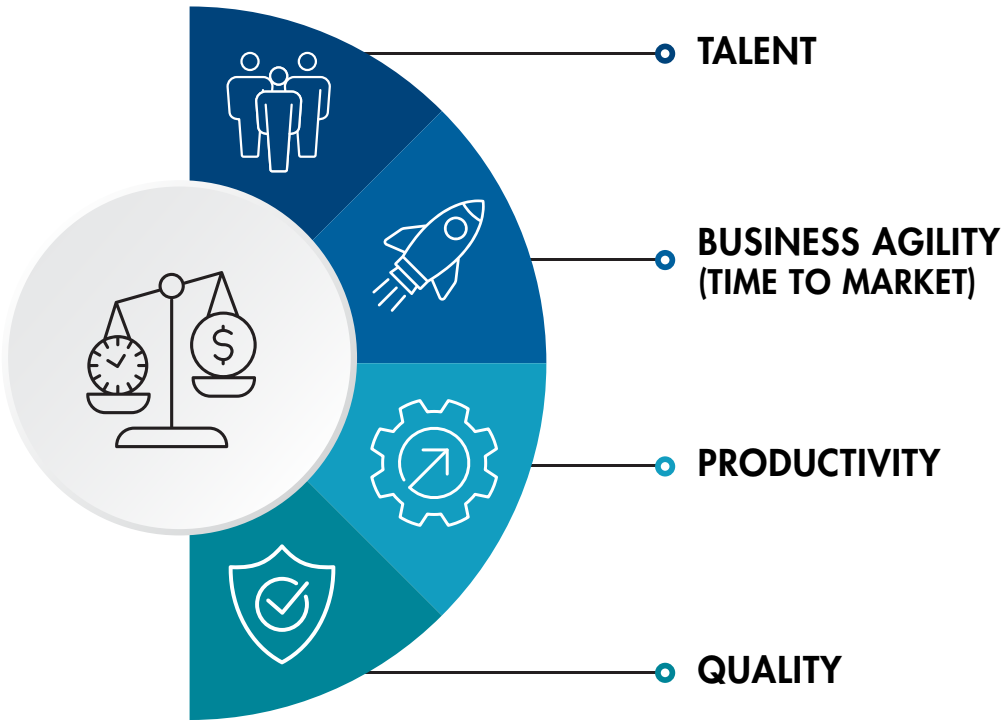
Basic value drivers are a set of foundational measures to benchmark and determine the CBA/ROI derived from AICGTs. There is some overlap between these value drivers and those enumerated as part of the industry value drivers in the next section. Carriers are encouraged to start with measuring the value of their AICGTs implementations by leveraging a combination of these fundamental drivers and incorporating measures from industry value drivers as is appropriate for the size, scope, and scale of your AICGT implementation for your firm. Contingent on the size of your AICGT implementation, getting started with measuring value might be as straightforward as adopting measures listed under basic value drivers (BVDs).These are listed in the grid below, along with why (and how) they should be measured.

Basic Value Driver (BVD)	Why	How
Productivity	AICGTs accelerate software development and programming coding tasks and reduce time invested in authoring code.	• Lines of code generated by AICGT vs. manually authored • Reduction in time for programming per feature • Autocomplete percent of code used in IDE
Time-to-Market Acceleration	AICGTs expedite the software development lifecycle, enabling quicker software release turnaround times ergo delivering business value sooner.	• Reduction in software development time • Time span from when a feature is requested to deployment in production • Sprint velocity (increase in sprint velocity for projects running Agile Scrum)
Quality and Reliability	AICGTs reduce errors in code and defects by suggesting best practices and reducing human errors.	• Defect reduction rate • Automated unit test pass rates • Reduction in post go-live defects (typically categorized and measured as critical, high, moderate, and low severity defects.
Talent Attraction, Satisfaction & Retention	AICGTs improve workflows by handling repetitive coding tasks, reducing frustration and burnout. As AICGTs become more ingrained and common across the industry, not having an AICGT as a key part of your standard development toolkit might deter future applicants and turn into a talent attraction issue.	• Developer surveys on AI tool satisfaction (eNPS for AICGTs) • Retention rate of developers using AI tools • Reduction in onboarding time for new developers • Seniority levels of talent using AICGTs • Whether AICGTs feature in hiring/job descriptions
Standardization	AICGTs enforce programming best practices across teams, ensuring uniformity in large-scale projects, including agile and scaled agile initiatives.	• Reduction in inconsistencies • Compliance with internal software development standards • Adoption rate of AICGT suggested code
Cost Savings	Over time, it is likely that AICGTs will allow firms to increase their throughput with the same number of FTEs. Moreover, there are direct cost savings that will result from the increase in quality and mitigation of human errors by avoiding expensive defect resolutions.	• Reduction of development cost per feature • Reduction in external contractor costs
Improved Documentation	Developers are sometimes loathe to document code. This is especially true during times of duress when they need to complete tasks or stories by a specific date. Over time, this can result in unmaintainable or unmanageable code. Knowledge transitions become extremely challenging, and this institutionalized knowledge can contribute to the buildup of legacy systems and pose a challenge for modernization activities. AICGTs help with code documentation, and this improves knowledge transfer and maintainability.	• The percentage of code comments generated automatically • Developer feedback on quality of comments and documentation
Secure Coding Practices	AICGTs enforce cybersecurity and information security best practices in code suggestions, resulting in more secure software systems.	• Security exposure detection rates • Compliance with secure development standards
Maintainability	AICGTs assist in reducing technical debt by allowing for the refactoring and optimization of code.	• The percentage of AICGT-assisted refactoring • Reduction in code complexity (consequent increase in understandability and maintainability)

2.2 Industry Value Drivers

Industry value drivers are an inventory of measures that were identified by the LIMRA and LOMA AI Industry Group: the AI Governance Group (AIGG). This inventory will encompass tangible and intangible standards of measuring value from AICGTs. The focus of the AIGG AI Code Generation Tools Value Drivers was to identify what should be measured with AICGTs, how to measure and quantify value, what benchmarks should be established or leveraged by the AIGG, etc. Most firms have templatized CBA and ROI frameworks in place for their organizations. With these value drivers having been identified and prioritized, it is expected that they will serve as a **starting point** for firms to incorporate into their CBA/ROI templates **(including the CBA/ROI templates developed by the AIGG)**.

These value drivers are classified into four categories: business agility (time to market), productivity, quality, and talent, as depicted in the figure below:



NOTE THAT VALUE DRIVERS WITH A BLUE BOLT ARE PRIORITIZED VALUE MEASURES —THAT IS, CONSIDER LEVERAGING THEM FIRST IN YOUR CBA/ROI

2.2.1 BUSINESS AGILITY (TIME TO MARKET) VALUE DRIVERS

- Reduced time for writing and executing unit and integration tests
- Tying code generated to agile stories — percent of stories automated or partially automated?
- Reduced time on non-engineering tasks (documentation, change requests, status updates)
- Amount of time saved
- Amount of throughput
- Increased time to market
- Frequency of deployments increase/decrease
- Number of stories per sprint increase/decrease
- DORA Metric: Lead time for changes/Did days from story start to production decrease?
- Reduced time spent refactoring code
- Reduced contractor dependency

2.2.2 PRODUCTIVITY VALUE DRIVERS

- Amount of code recommendations from AICGT being accepted (not just lines of code, but functionality)
- Reduced feature/story delivery time
- Lines of code generated (metric available from within the tool itself)
- Suggested lines of code accepted (metric available from within the tool itself)
- How much net new code is being generated
- How much code refactoring AICGT produced versus typical years with manual refactoring
- DORA Metric: Mean time to recovery (time to resolve production issues; volume of production issues/bugs can also be used)
- Cost Metric: Cost per feature
- Cost Metric: Cost per release (AI tools make our delivery more efficient, should drive cost down)
- Volume of feature/story delivery

2.2.3 QUALITY VALUE DRIVERS

- Amount of value that the accepted recommendations generated (i.e., If developers are accepting 30 percent of the recommendations, how much value did that create? What yielded in that 30 percent?)
- Software QA via automated testing (number of unit tests, code coverage)
- Reduced code vulnerabilities
- Reduction of errors
- Cleaner code
- Safer code
- More performant code
- Reduction in errors and bug fixes across all stages of testing
- Adherence to standards and best practices
- DORA Metric: Production change failure rate

2.2.4 TALENT VALUE DRIVERS

- INTANGIBLE MEASURE: How many people are satisfied with the tool (measured via instrument such as a pulse survey)
- INTANGIBLE MEASURE: Job satisfaction: Successful reskilling of programmers to new languages
- Measure on how quickly and effectively reskilling engineers to programming language they aren't skilled in:
    - How much time did we save?
    - How much more throughput did we have?
    - Measure of if and how developers are getting upskilled with this because the AI is teaching them better ways to accomplish tasks
    - Increased time to market
    - Number of software engineers that successfully learn a new programming language
    - Time required for a software engineer to be effective with a new programming language
    - Time required for a software engineer to have knowledge parity with a new programming language
- How many people are using the tool (adoption)
- What is the seniority of the talent using the tool/s
- INTANGIBLE MEASURE: How do developers feel about these tools (eNPS measure)?
- INTANGIBLE MEASURE: Would developers recommend AI code generation tools to other developers (NPS)?
- Rate of new developer onboarding change (especially entry-level roles)/Were they more productive sooner?

2.2.5 BENCHMARKS BEING CONSIDERED

- Where adoption is strongest (mainframe vs. newer technologies)
- What languages are these tools being used for (Java, .NET)?
- What languages are these tools better suited for?
- Training provided to developers on the tools (i.e., prompt engineering?)
- User population using tools
- Tools being used beyond code generation — architectural diagrams, workflows, etc.
- Unit testing with Copilot
    - What it is good at vs. not?
    - Basic unit testing vs. mocks
- Churn that the AI development tools can create and how do we educate to prevent that?
    - Developers utilize GitHub Copilot to create/stub out a set of code for a specific problem
    - Developers know what they want
        - Copilot does not generate what they want specifically, so they continue to modify the prompt to get the output that they are expecting
        - The next result is that they spent more time tuning the prompt vs. writing the original code

APPENDIX A: CI/CD Pipelines

Continuous integration (CI) is a software development practice where developers frequently merge their code changes into a shared repository. CI helps identify defects earlier in the process because the code is automatically tested every time it's updated, resulting in fewer unexpected surprises and issues down the line.

Continuous deployment (CD) is an extension of CI where the code changes are automatically deployed to your production environment after passing all requisite tests. CD ensures that new features, enhancements, and defect fixes are delivered quickly and reliably without manual intervention.

CI/CD allows for a much faster time to market, delivers software of higher quality, increases efficiency, and helps firms deliver better software faster and more efficiently. A CI/CD pipeline is analogous to a manufacturing assembly line in that as products move through different stages of assembly, software changes move through various stages (code integration, then automated testing, then deployment) in the CI/CD pipeline.

LEARN MORE about the AI Governance Group (AIGG) **here**.

Advancing the financial services industry by empowering our members with

LIMRA LOMA
Navigate With Confidence

0195-2025