INSTANT PAPER

Copilots for Coding

Exploring the impact of GenAI on software development.

© engineering

Authors

Massimo Del Vecchio
Eng Modernize
Executive Director

ENGINEERING
Massimo.DelVecchio@eng.it
in Massimo Del Vecchio

Dario Fabbri
Modernization Factory
Senior Technical Manager

ENGINEERING
Dario.Fabbri@eng.it
in Dario Fabbri

Elena Marchisa
Strategic Marketing
& Content Senior Specialist

ENGINEERING
Elena.Marchisa@eng.it
in Elena Marchisa

© engineering

00

Summary

01 / **GenAI on software development** 2
02 / **Our approach to GitHub Copilot Trial** 3
03 / **The productivity gain** 6
04 / **The software quality improvement** 9
05 / **Toward broader adoption** 12
06 / **Why choose us** 13

© engineering

01 GenAI on software development

Generative Artificial Intelligence (GenAI) enables the creation of content - code, images, audio, and video - that not only replicates but often exceeds human capabilities in speed and precision. Companies have found in Generative AI a powerful tool for optimizing and improving efficiency in multiple business areas.

In the field of **software development**, Generative AI is revolutionizing the traditional approach to code From simple tools providing hints to content and code creation, this technology is redefining every phase of the software lifecycle: requirements gathering, design, development, testing, and deployment, as well as documentation production. Even though we are only at the beginning, the future looks promising. As this technology continues to evolve, developers are increasingly turning to GenAI tools to **streamline software production lines, enhance productivity, and improve code quality**.

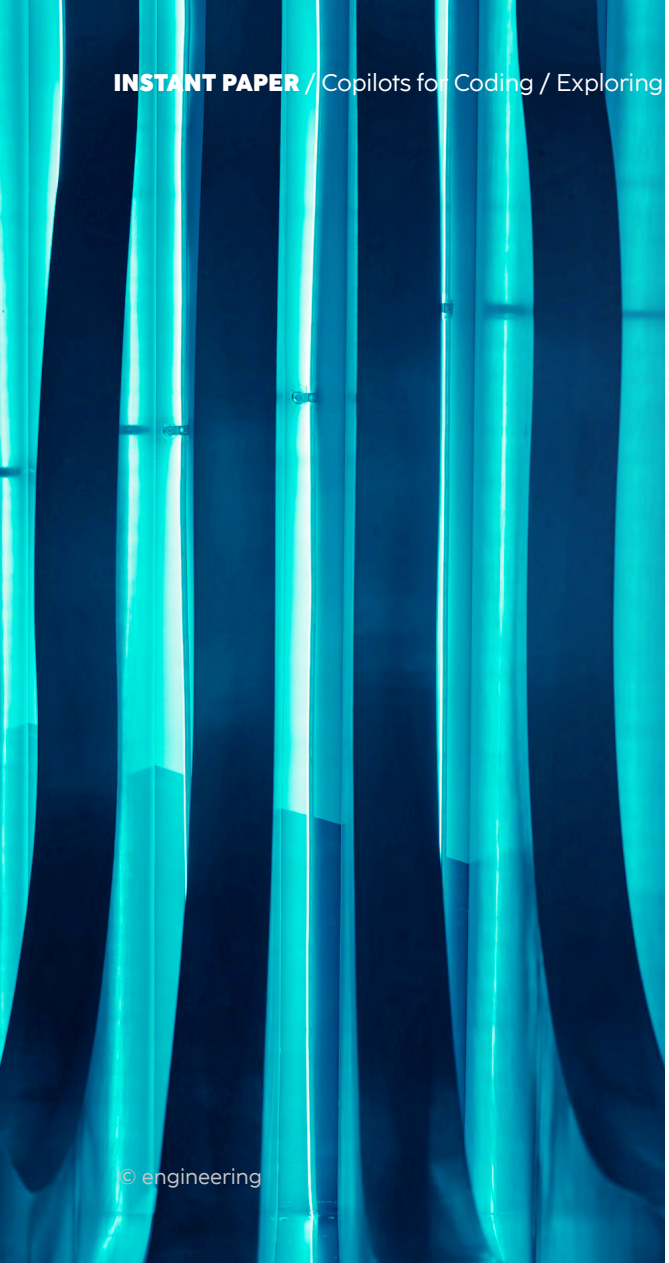
Code copilot systems, such as GitHub Copilot, powered by Large Language Models are among the most used tools. These systems act as intelligent assistants, capable of generating code, suggesting improvements, and even automating entire steps of productivity processes.

This paper will present the **GitHub Copilot adoption trial** we held across different development teams in real business projects and our key findings on it.

Our results clearly demonstrate on the field the benefits GenAI is bringing into software development.

The successful integration of GenAI in development environments marks a significant milestone in the ongoing digital transformation, leveraging the peculiarities of human-machine collaboration to create more efficient, scalable, and innovative software solutions.

© engineering



© engineering

02

Our approach to GitHub Copilot Trial

GitHub Copilot provides contextualized assistance throughout the software development lifecycle, from code completions and chat assistance in the IDE (Integrated Development Environment) to code explanations and answers to docs in GitHub and more.
We chose to use Copilot for our trial because of our strong partnership with Microsoft and the fact that, as of January 2024, it was the only AI assistant able to integrate the chat service into the most adopted IDE (Jetbrains and Visual Studio).

Our journey with GitHub Copilot began with a clear mission: **understand the actual impact of AI on developer productivity and software quality**.

Unlike lab-based studies, which often fail to capture the nuances of real-world projects, we chose to conduct our trial under authentic business-as-usual conditions, randomly selecting from existing ongoing projects and defining a simple but effective approach for our context.

We first defined our **strategy** to collect data and measure productivity. As System Integrators, our software development and productivity measurement methodologies

Copilot adoption

TRIAL POPULATION

9 MONTHS | **70+** DEVELOPERS | **10+** TEAMS

Types of project

- Green field
- Brown field
- Application maintenance
- Portfolio handovers
- Modernization
- Security & quality remediations

Technologies

- Java legacy
- Java microservices (Spring Boot / Quarkus)
- .NET legacy
- .NET microservices
- Angular / Typescript
- ...

Seniorities

- Technical managers
- Team leaders
- Architects
- Senior developers
- Junior developers

can be very different across multiple clients and contexts: ranging from Function Points, Story Points, and Lines of Code to no measurement framework at all. We had a clear "scattering problem" on delivery methodology.

We developed a **cross-team standard approach** to allow to test benefits of GenAI in software development in any business context.

It consists in three subsequent steps:

1. Categorization of Development Activities
A shared taxonomy of task types is developed to align all developers with consistent definitions.

2. Manual Time Estimation
Before writing code, it is asked developers to estimate the time (in man-hours) required for specific tasks. They are instructed to try and measure small/medium sized activities (hours not days), in order to reduce estimation biases and errors.

© engineering

3. Comparison and Adjustment
The estimated time is compared with the real time spent on tasks accomplished with Copilot. A correction factor is applied to adjust for contingencies on the initial estimates also through direct interviews with developers.

With reference to the development activities categorization, in our trial we divided the tasks into specific areas designed to bring out the strengths of the Copilot:

- **Quick & Dirty Feature Development**: the creation of initial version of a functionality to meet basic functional requirements. No comments, logging statements, automatic tests and optimizations are included.
- **Code Analysis**: all the activities related to understanding existing codebases.
- **Refactoring**: the restructuring or rewriting of the code to improve performance, quality, readability and maintainability without altering functionality.
- **Logging**: adding or enhancing logging statements to existing code for system monitoring and troubleshooting.
- **Test Automation**: the creation or improvement of any form of automated tests. No distinctions are made between unit tests, integration tests, end-to-end tests, etc.
- **Technical Documentation**: the draft of technical notes for code blocks to streamline documentation processes.
- **Comments**: the writing or improvement of source code comments to enhance maintainability.
- **Hotfixes**: the buf-fixing activities in released code under tight deadlines.

The trial was divided into **three waves** of three months each one, starting from January 2024, to have a sufficiently large amount of time to collect significant data.

Before entering the third wave, we decided to make a **large reshuffle** in the trial population. This allowed us also to reevaluate the developers' learning curve and verify the data collected in the second wave.

This process aimed to monitor Copilot onboarding on a project-by-project basis to maximize the effectiveness of the investment.

© engineering

03 The productivity gain

The collected data demonstrated Copilot's effectiveness across diverse tasks, with measurable improvements in both productivity and quality.
Below we present the results of the **first** (January, February and March 2024) and the **second wave** (April, May and June 2024). Data from the third wave is still being consolidated as of the date of this writing.

Each task estimation has been collected across the wave and named "Data Point".

The estimated productivity increase was calculated as a percentage adopting with the following formula:

(Estimated time without copilot - effective time with copilot) / (Estimated time without copilot) *100. Productivity increases have been plotted across a Gauss distribution for all categories of development activities.

The following charts show the actual distribution, where:

- The **Data Points** value indicates the number of observations (single development task) for the wave of reference;
- The **value "μ"** represents the **mean** value for the metric during the observation period;
- The **value "σ"** represents the **Standard Deviation**. A higher σ indicates more variability, i.e. how much the Data Points are far from mean value;
- The **curves** shown in the chart are a probability distribution of data around the mean. The **dashed line** represents the values of the first wave, and the **solid line** represents those of the second one.

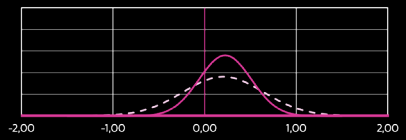
For example: the mean value of 22.28% on Quick and Dirty features means that the time spent using copilot is 22.28% less with respect to not using it (with a standard deviation from the average of 28.46%).

© engineering

Productivity results of wave 1 and 2

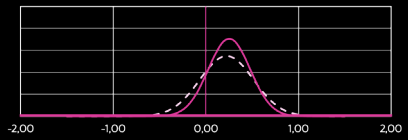
Quick & dirty feature dev

Data points	55	**102**
μ	20.15%	▲ **22.28%**
σ	43.96%	▲▲ **28.46%**



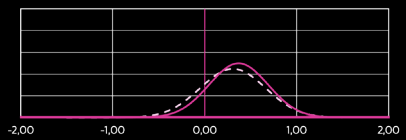
Refactoring

Data points	29	**54**
μ	22.58%	▲ **25.36%**
σ	29.08%	▲ **22.60%**



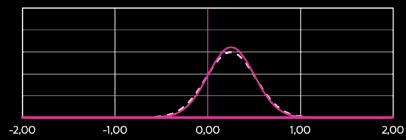
Test automation

Data points	20	**34**
μ	30.63%	▲ **36.83%**
σ	35.44%	▲ **31.97%**



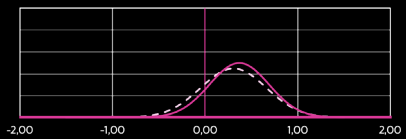
Comments

Data points	19	**30**
μ	25.00%	≈ **25.36%**
σ	26.69%	▲ **24.76%**



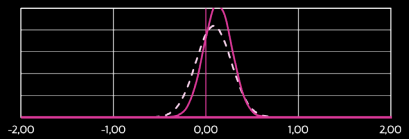
HotFix

Data points	4	**24**
μ	16.67%	▼ **11.06%**
σ	34.74%	▲ **14.03%**



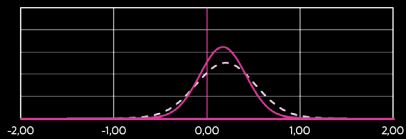
Code analysis

Data points	8	**9**
μ	8.33%	▲ **12.96%**
σ	19.01%	▲ **15.20%**



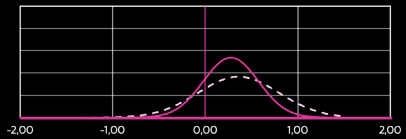
Logging

Data points	5	**7**
μ	20.00%	▼ **17.01%**
σ	31.84%	▲ **24.72%**



Technical documentation

Data points	5	**7**
μ	36.19%	▼ **27.89%**
σ	43.72%	▲ **29.67%**



▲▼ : improvement or worsening of the value

© engineering

During wave two, the standard deviation has been reduced approximately of a **10%** across multiple coding categories. This data may be the result of both the developer learning curve in tool adoption and the improvement of their activities estimation capability.

Among the reported improvements, **7-10% directly refer to a productivity boost**, due to Copilot's real-time suggestions and explanations (e.g.: code commenting has been classified as quality improvement, not productivity).

This increase might appear quite big, but it only refers to the hours the developers really spend on coding during the whole IT project lifecycle. A developer's workday, in fact, includes various non-coding activities such as meetings, document reviews, and training. Even on a very focused day, coding does not overcome 70% of the actual working time. Moreover, it only represents about 30-40% of the total effort in a typical IT project.

Finally, in our trial we analyzed a very large, but non complete subset of development activities (e.g.: we have not evaluated overall architecture definition and implementation, task breakdown, ecc.).

To summarize, GitHub Copilot helps the developers only when they have their hands on the keyboards and their IDEs open so the overall impact on project performances must be calculate across the whole delivery stack and requires further deepening.

Lastly, mention should be made of a parallel trial, conducted during waves 1 and 2 in which some of the teams were also involved in remediation activities related to issues detected by the SonarQube static code analysis tool.
The use of Copilot for Sonarqube remediations brought an average **productivity boost of 40%**. This highlighted the **great potential to fully automate repetitive and low creative tasks into daily coding activities**.

© engineering

04 The software quality improvement

© engineering

Copilot enabled better adherence to best practices, reduced bugs, and enhanced code maintainability, thus resulting in about **10% improvement** also in software quality.

The metrics provided by GitHub Copilot (last usage, number of proposed and accepted suggestions, ...) proved to be very helpful in understanding the engagement of the group, but we also conducted a direct survey with developers involved in the trial for further investigation.

Generally, there has been a good answer to the use of GitHub Copilot. Developers tend to be technology enthusiasts, and the largest part of the population has been eager to try GenAI, which they perceive will have a transformative role in their professional future.

Junior developers have emphasized that they may be prone to hallucination errors.

Senior profiles highlighted the need of experience to better evaluate the suggestions proposed by Copilot but admits it is cleverer in crafting code with very specific and effective prompts.

- **Code Analysis** ★★★★★
- **Refactoring** ★★★★
- **Logging** ★★★
- **Test Automation** ★★★★
- **Technical Documentation** ★★★★★
- **Comments** ★★★★
- **Hotfixes** ★★★★

GitHub Copilot shows promise in its ability to transform software development practices, but its utility varies depending on the task at hand. Our experimentation highlighted some of its current limitations.
When asked to generate **documentation** for existing codebases, the Copilot provides basic documentation for simpler or well-structured code, but its performance diminishes with more complex or poorly written legacy code, where it struggles to infer intent or identify nuanced logic.

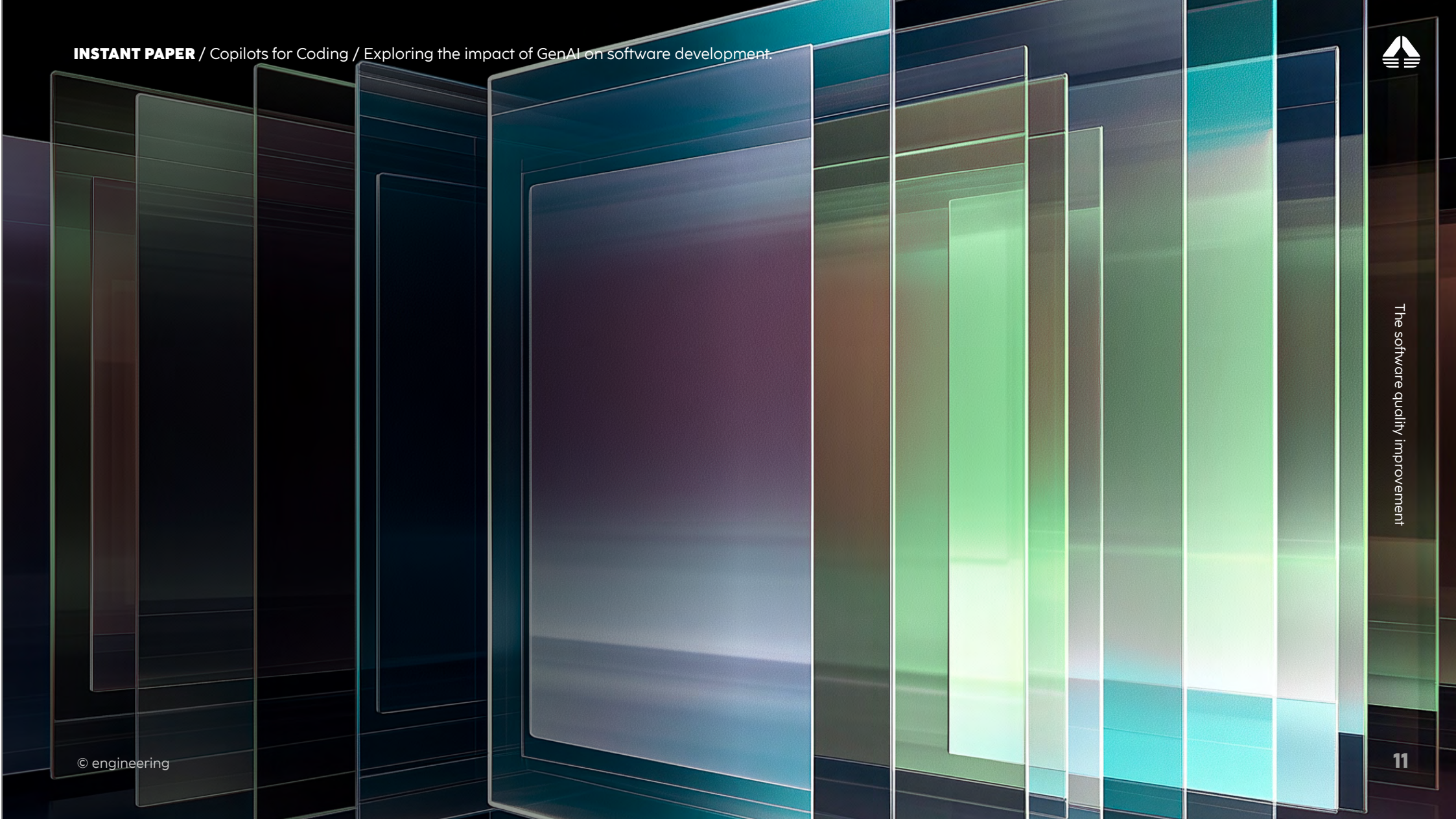
In **Code Translation**, initial tests for translating code from very old languages, such as COBOL, to modern ones have shown poor results: as of today, the tool struggles with COBOL due to its verbosity, procedural nature, and highly domain-specific context, producing inaccurate translations. Similarly, for frameworks like Spring Boot, while it can assist in basic refactoring or setup, its understanding of deeper architectural patterns and framework-specific best practices is limited.

With reference to COBOL and code documentation, as Engineering Group we can instead boast of a successful project using our Private GenAI Assistant, EngGPT.

Returning to the **contextual challenges**, Copilot's lack of understanding of the domain-specific context, coupled with the inherent complexity of older technologies, reduces its ability to provide full assistance in modernization projects.

These limitations highlight the need **for significant human involvement** in such translation and modernization efforts and the importance of experienced developers in leading efforts to transform legacy systems.
While the future looks promising, considering the market hype and the large-scale investments expected on the "GenAI for coding" field, currently copiloting development activities with GenAI provides improvements for well-defined and repetitive tasks.

© engineering



© engineering

05 Toward broader adoption

Our trial suggests that Copilot works well when **paired with experienced developers** and, with correct supervision, can support junior people to speed up their learning path on software development. In general it **boosts velocity for repetitive tasks** for all kind of developers.
GenAI is for sure a time creation tool, allowing the developers to concentrate on higher value activities.

For now, Copilot may assist in **preliminary analysis** or low-stakes tasks in legacy code modernization efforts, but it is not ready to replace manual work in these domains. Further areas of investigation may concern the code churn, combining Copilot metrics with GitHub data. It would be necessary also to explore Copilot support in requirements gathering, software documentation, reverse engineering, (semi)automatic conversions.

As GenAI on coding will grow, improvements in its **ability to handle legacy code and complex documentation** tasks may emerge, particularly if future iterations are trained on **larger and more diverse datasets** that include older programming paradigms and frameworks. At the status of the product, we do not believe that the goal of GenAI adoption is to produce tons of unchecked code as fast as possible. As of today, we think GenAI can led to significant quality improvement (unit testing, comments, refactoring, tech doc, ...) more than productivity itself. Anyway, we see this as the beginning of a much larger transformation, driven by significant investments in GenAI technologies across the IT market. In a scenario characterized by an abundance of proof of concept and research on GenAI, the use of new technologies in industrialized projects allows us to clearly identify the way forward to make a real impact in the world in which we live and work in.
The latter consideration led us on taking the bold decision to **distribute GitHub copilot to all our developers into Eng Modernize**: only with a full industrialized approach we will be able to explore the full potential of GenAI products into software development. The plan aims to reach thousands of developers across 2025, periodically checking results regarding quality and productivity improvements along the way.

In parallel to “industrial” distribution of the GitHub Copilot we are investigating other market tools to ensure we always adopt the right solution for the right use case.

The industrialized adoption confirms Engineering position as the strategic partner to onboard GenAI’s services on software development, bringing efficiency and overall improvement to all kinds of businesses.

© engineering

06 Why choose us

© engineering

Eng Modernize

Transform the clients' ecosystems into a fully digital environment, leading and supporting them in their Journey to cloud

- Cloud native solutions deployed on top vendors in the market: AWS, Azure, Google.
- On prem and private cloud solutions implemented on «cloud ready» technologies.
- A Cloud Modernization Process to guide the customer through the cloud transformation.

10+
Years of expertise in **cloud native** projects

DISTRIBUTED FACTORIES
With on-shore and near shore capabilties

BILLIONS
Of daily transactions on our cloud assets delivered to our customers

1500+
Developers specialized on cloud native solutions and modernization of enterprise software

BILLIONS
Of lines of code where we are running GenAI solutions to improve quality and productivity

CLOUD DELIVERY IN MULTIPLE MARKETS
Government, Industry, Energy & Utilities, Media & Communication, Transportation, Finance

Specialized factories

(EAI - Enterprise Application Integration - and GIS - Geographic Information System)

Advisory | **Cloud native implementations** | **Deploy automation** | **PAAS services**

Data displayed represents our elaboration of data coming from multiple sources

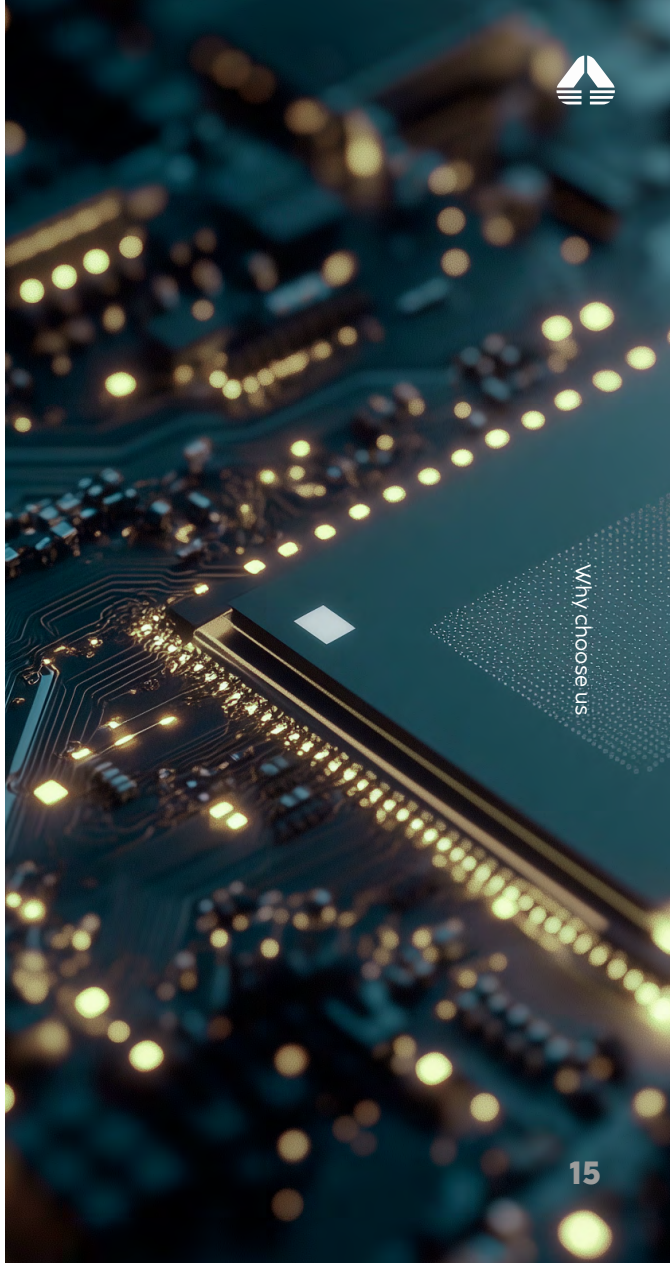
© engineering

Legacy systems could limit business growth and require the adoption of faster and more agile solutions. **Cloud-native architectures**, supporting scalability and flexibility, enable the development of modern applications with faster time-to-market, fueling business innovation and efficiency.

Eng Modernize is our Technology Business Line within **Digital Technologies** Business Unit with significative experience in large migration projects and strong technical capabilities transforming legacy application into scalable and secure solutions.

We manage the **full cycle** of modernization process, sharing **best practices and frameworks**, offering an advisory-led approach, to ensure smooth transitions and optimized cloud and on-prem solutions. We deliver **tailored and customized outsourcing services** to optimize resources, mitigate risks and reduce costs, also through established partnerships with top vendors. In Eng Modernize, we leverage Generative AI for 4 main goals:

- **improve quality**, reducing maintenance costs and allowing for easier evolution of the codebase with specific offering on test automation;
- **reduce bugs** since the early detection of potential problems greatly reduces the costs typically associated with solving bugs in production;
- **improve security**, both for compliance with regulations and for limiting the risks of damage for the company and its customers;
- **improve productivity** by reducing the time spent on more trivial tasks can free development resources for higher-value activities.



© engineering

www.eng.it

Engineering Group

@LifeAtEngineering

@EngineeringSpa