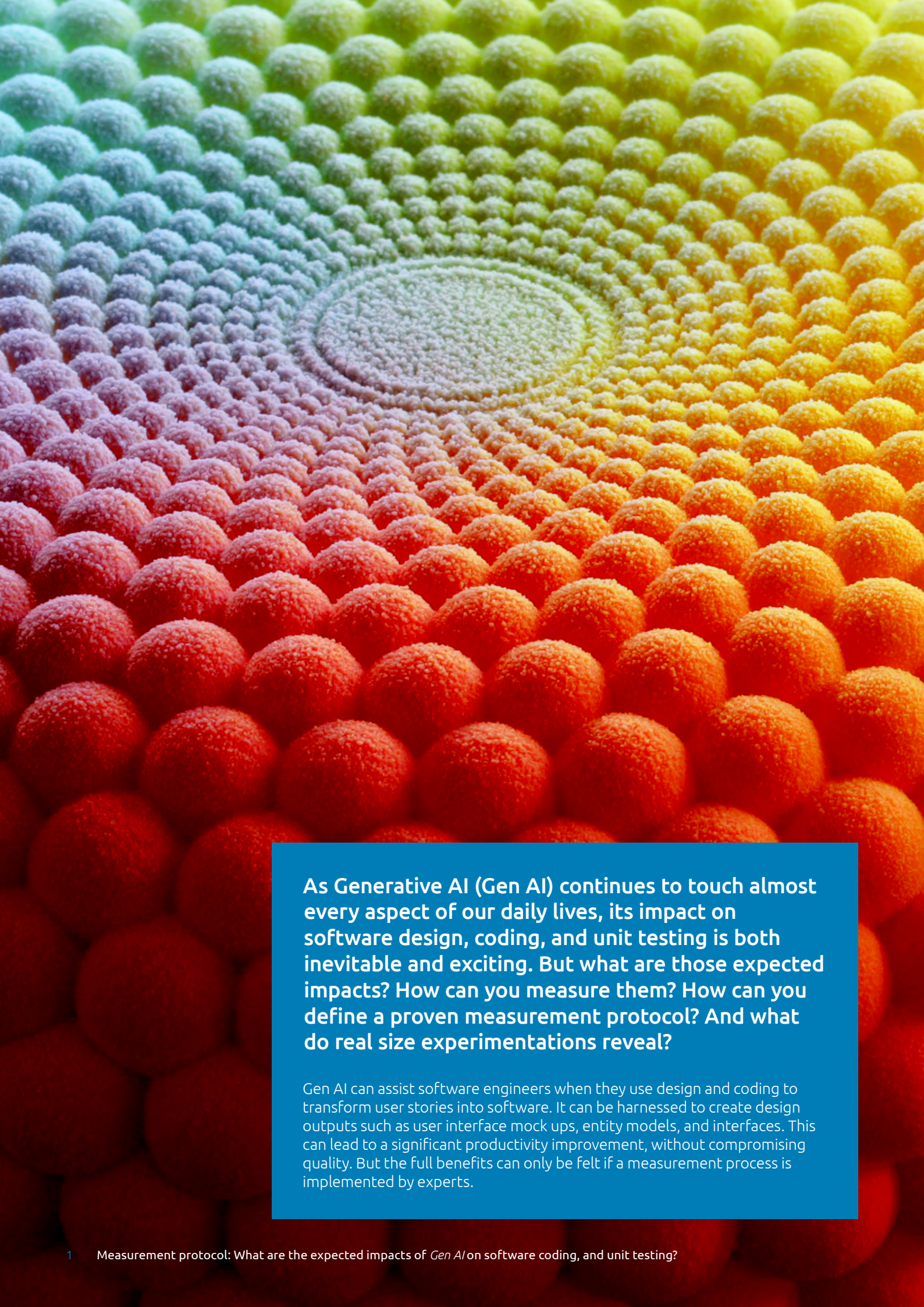
How to measure the impact of *Gen AI* on software coding and unit testing?

Capgemini

CONTENTS



As Generative AI (Gen AI) continues to touch almost every aspect of our daily lives, its impact on software design, coding, and unit testing is both inevitable and exciting. But what are those expected impacts? How can you measure them? How can you define a proven measurement protocol? And what do real size experimentations reveal?

Gen AI can assist software engineers when they use design and coding to transform user stories into software. It can be harnessed to create design outputs such as user interface mock ups, entity models, and interfaces. This can lead to a significant productivity improvement, without compromising quality. But the full benefits can only be felt if a measurement process is implemented by experts.

How will *Gen AI* impact software programming?

First of all, what impact will Gen AI have on software programming, according to businesses and organizations? Our latest Capgemini Research Institute[1] report shows that 61% of organizations see enabling more innovative work, such as developing new software features and services, as the leading benefit Gen AI. Close behind are improving software quality (49%) and increasing productivity (40%). Organizations are utilizing these productivity gains on innovative work such as developing new software features (50%) and upskilling (47%). Very few aim to reduce headcount (4%).[2]

Gen AI is poised to redefine conventional programming practices by shifting the focus from coding to prompt engineering and code proofreading. This is expressed perfectly by Andrej Karpathy, an OpenAI computer scientist, who recently said: "the hottest new programming language is English[3]."

What does this mean in practice? As an example, software engineers can use plain language to describe the intended functionality of a software feature, then review, update, and validate the generated output. There are many other examples, such as auto-completion of code, generating code for unit testing, (retro) documentation, and code migration from one language to another. Of course, Gen AI is already valued by developers because it supports them during coding. It can either suggest clean code directly or evaluate existing code to improve software quality if it identifies issues.

Software quality can be traced back to the early testing phase, where the unit test cases and / or related test data sets fail to include the full spectrum of possible user inputs and scenarios. Gen AI can assist developers in writing more complete unit test cases, in which user stories provide prompt engineering context for maximum relevance. It can generate a massive amount of synthetic information that closely resembles real world data to ensure high unit tests coverage.

Although adoption of Gen AI for software engineering is still in its early stages with 9 in 10 organizations yet to scale, 27% of organizations are running Gen AI real size experimentations, and 11% have started leveraging Gen AI in their software function. Gen AI is expected to play a key role in augmenting software workforce with better experience, tools and platforms, and governance, assisting in more than 25% of software design, development, and testing work by 2026.[4]

[1] *Capgemini Research Institute "Turbocharging software", June 2024*
[2] *Capgemini Research Institute "Turbocharging software", June 2024*
[3] *https://twitter.com/karpathy/status/1617979122625712128?lang=en-GB*
[4] *Capgemini Research Institute "Turbocharging software", June 2024*

Why is measurement important?

In the fast-paced and ever-evolving landscape of modern technology, making informed decisions is crucial to success. However, extracting meaningful insights can be a daunting task in a world inundated with data. Therefore, establishing a measurement framework is essential. It serves as a navigational aid in a vast sea of information, guiding teams from raw data to actionable decisions.

Measuring the performance of Gen AI ensures that it meets the desired objectives, whether that's improving efficiency, enhancing accuracy, or reducing costs. It also helps identify areas for improvement, guiding further development and optimization. And it provides accountability, demonstrating the value and return on investment (ROI) to stakeholders.

At the same time, measurement allows us to quantify attributes, which enables us to compare, analyze, and understand things more effectively. It also allows for progress tracking and performance evaluation, and provides data-driven insights that inform decision-making processes.

Challenges in measuring *Gen AI* impact

"What gets measured, gets managed" might be an old saying but it rings true in the new paradigms in which Gen AI is unfolding. It's all well and good to implement, but measurement is crucial. However, measuring productivity is inherently complex due to the multifaceted nature of development work as part of the software development lifecycle (SDLC), the dynamic and evolving environment in which it occurs, and its inherent subjectivity and intangibility. Effective measurement requires a holistic approach that considers qualitative and quantitative factors, including context-specific considerations.

Measuring software quality is a challenge as it encompasses multiple dimensions, including functionality, performance, reliability, usability, maintainability, security, and scalability. Assessing quality requires considering these diverse aspects, each with its own set of metrics and criteria. Another challenge is that the different stakeholders have different priorities, whether they're customers, businesses, architects, developers, testers, or in operations.

Feedback from the software engineers who will be using Gen AI on a daily basis also needs to be considered. This is an important topic as Gen AI has an impact on the development environment and the way they work.

Nearly nine in ten (86%) of large organizations, with annual revenue greater than $50 billion, have adopted (piloted / scaled) Gen AI as compared to 23% of their smaller counterparts, with an annual revenue between $1-5 billion.[5]

[5] *Capgemini Research Institute "Turbocharging software", June 2024*

Establishing a measurement protocol

Now let's focus on how to define and implement a practical measurement protocol to get a clear view on the impact of Gen AI in coding, and unit testing as part of bespoke application development.

Almost half of organizations in our survey (48%) have no defined metrics to gauge the success of Gen AI use in software engineering. We also found that there seems to be no standard way of measuring productivity.[6]

Our survey reveals an important fact about commonly used metrics. While they're suitable for regular productivity measures, such as time to deploy or to resolve issues, they do not fully capture the benefits of Gen AI. Especially on non-conventional measures of productivity, such as employee satisfaction. These are better captured by metrics frameworks like DORA and SPACE.[7] However, DORA and SPACE are yet to gain traction, as they are costly and time-consuming to implement. This finding indicates that a set of metrics including KPIs for velocity, quality, security, and developer experience can prove useful.[8]

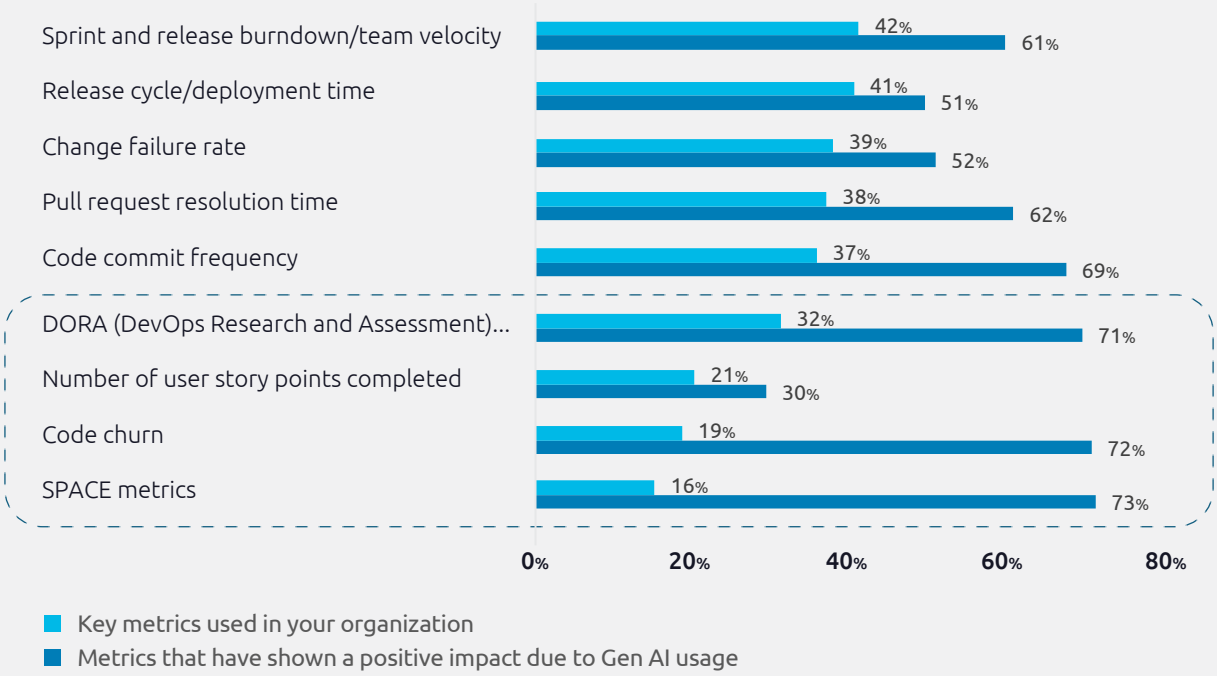
[6] *Capgemini Research Institute "Turbocharging software", June 2024*

[7] *DORA – through such metrics as lead time for changes, deployment frequency, mean time to recovery and change failure rate – measures how well an organization balances speed and stability. The SPACE set of metrics tries to comprehensively assess team dynamics and developer experience. It balances the assessment of technical output with the wellbeing of developers, which traditional metrics fail to do.*

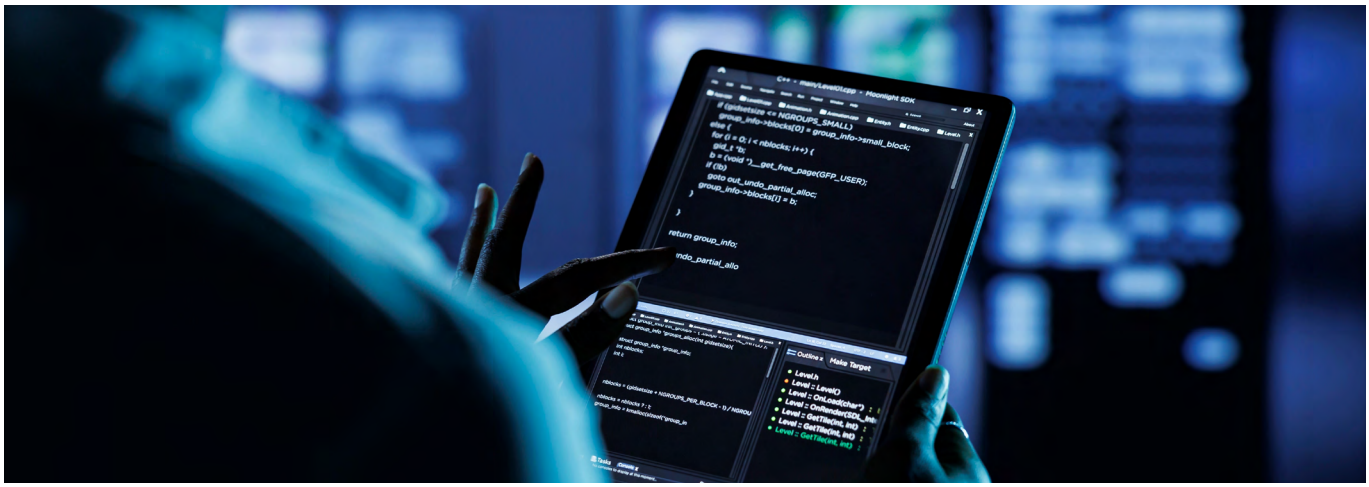
[8] *Capgemini Research Institute "Turbocharging software", June 2024*

Most organizations show improvement from use of Gen AI when measured using less popular, but more holistic, productivity metrics frameworks, such as SPACE and DevOps research and assessment (DORA):

Metrics predominantly used vs. Metrics impacted through use of *Generative AI*



The measurement protocol offers a well-defined process that creates comprehensible, comparable and reliable results.



Components – What does the measurement protocol consist of?

- **Teams organization:** Organize teams for significant and actionable results, using different patterns such as parallel teams, shadow teams, or multi-pyramid teams
- **Measurement approach:** Establish the timeline and the process for conducting the measurement, including the preparation, the baseline, and the execution.
- **Measurement metrics:** Identify the metrics for measuring the impact of Gen AI on software engineering, such as coding velocity, code quality, code security, and developer experience.
- **Measurement tooling:** Agree the tools for collecting and analyzing the metrics, such as SonarQube, CAST, Jira, or developer surveys.
- **Measurement reporting:** Select the templates and formats for presenting and communicating the measurement results, both in detailed and executive levels.
- **Prerequisites and success factors:** Put in place the conditions and factors that need to be met and considered for a solid and consistent measurement, such as team stability, duration, backlog, technology, tooling, legal, and cybersecurity.
- **Normalization process:** Define how to manage instability and variability during the experimentation execution, such as changes in team size, capacity, or complexity, and how to adjust the metrics accordingly.
- **Qualitative feedback:** Create a mechanism to harness information on the holistic experiences of developers in the form of surveys or verbatim reports. Be aware that negative experiences often provide the best learnings.

Metrics - What data will be evaluated during the real size experimentations?

Besides establishing and enabling the components it's essential to define the metrics that will be evaluated during the real size experimentations. These are the fundamental data points to be measured and analyzed.

- **Velocity:** This should be measured on different levels, as it's the most important metric.
  - Coding velocity: This is the key indicator of a team's productivity (coding and unit testing) and is typically measured in terms of implemented story points.
  - Coding velocity per developer capability / seniority: Calculated as the time taken to complete "X" story points with and without Gen AI assistance as per developer capability (good, average, below average)
  - Coding velocity per complexity: Calculated as user story points required with and without Gen AI assistance as per story complexity (simple, medium, and complex)
- **Unit testing coverage:** Essential metric for assessing the quality and reliability of software. To keep it simple we focus on instruction coverage (C0) as this is measured by most of the tools.
- **Code efficiency:** Measures the potential performance and scalability bottlenecks in software. To keep it simple we focus on static code analysis and not runtime analysis (for example, with profilers). This is not an industry standard, but a metric our clients find invaluable.
- **Code security:** Determines the risk of vulnerability issues and probability of breaches for an application.
- **Code smells:** Refers to indicators of poor or problematic code that may require attention or refactoring.
- **Code duplication:** Highlights the presence of identical or similar code fragments in different parts of a codebase. With Gen AI it's more likely to create code duplicates.

Team organization – What works best?

Executing the measurement protocol requires an engagement with a proper user story backlog.

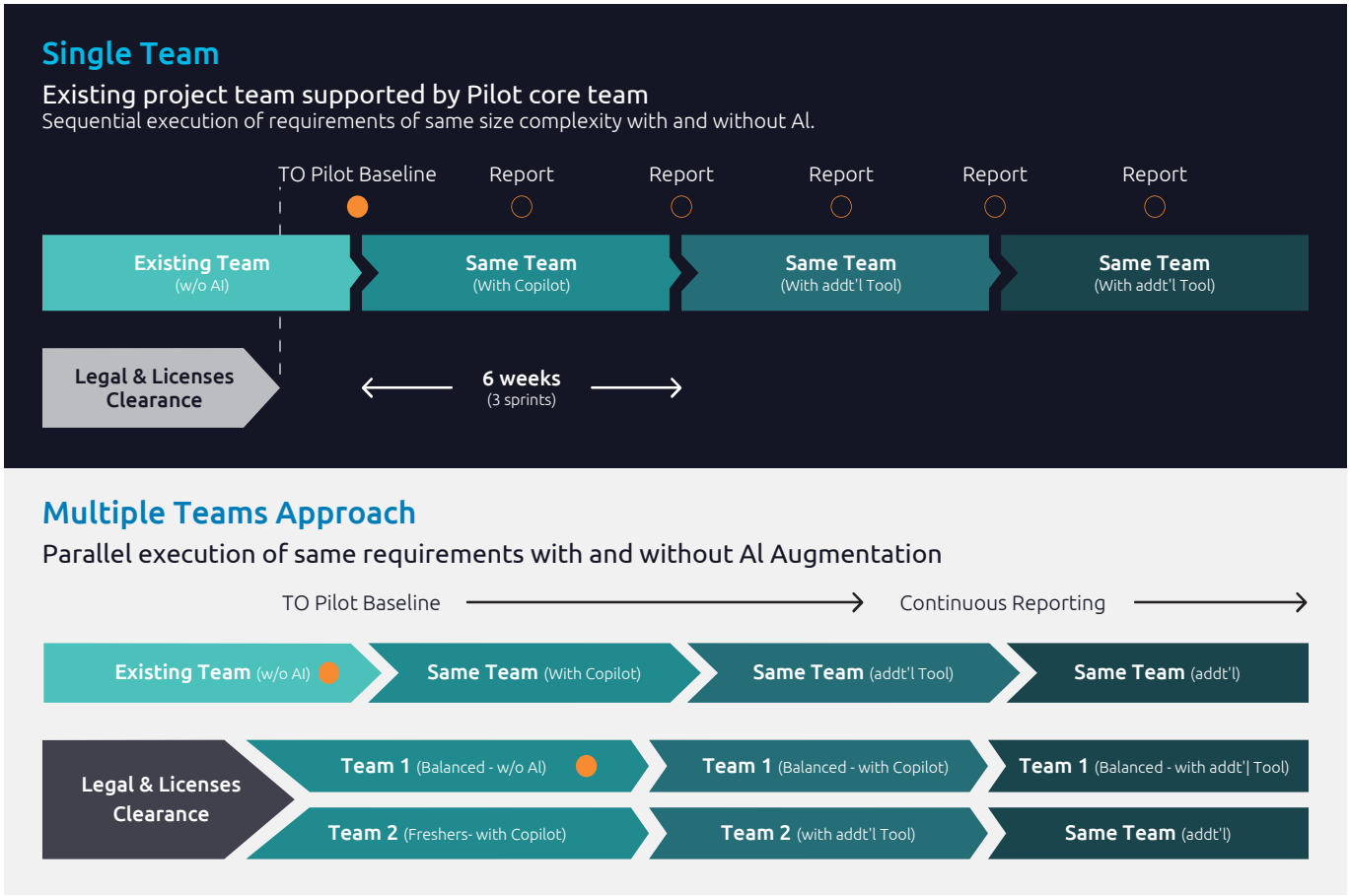
But how many teams should be involved in the measurement process? In our view, it's always better to have multiple teams to minimize the human factor.

- **Single Team:** Sequential execution of backlog of same user story size / complexity with and without Gen AI assistance by one team.
- **Multiple Teams:** Parallel execution of same backlog with and without Gen AI assistance by at least two teams.

Team setup - What is the optimal mix?

The seniority or capabilities of a team are important for normalization. Therefore, it's mandatory to know what kind of team mix is working on the defined backlog.

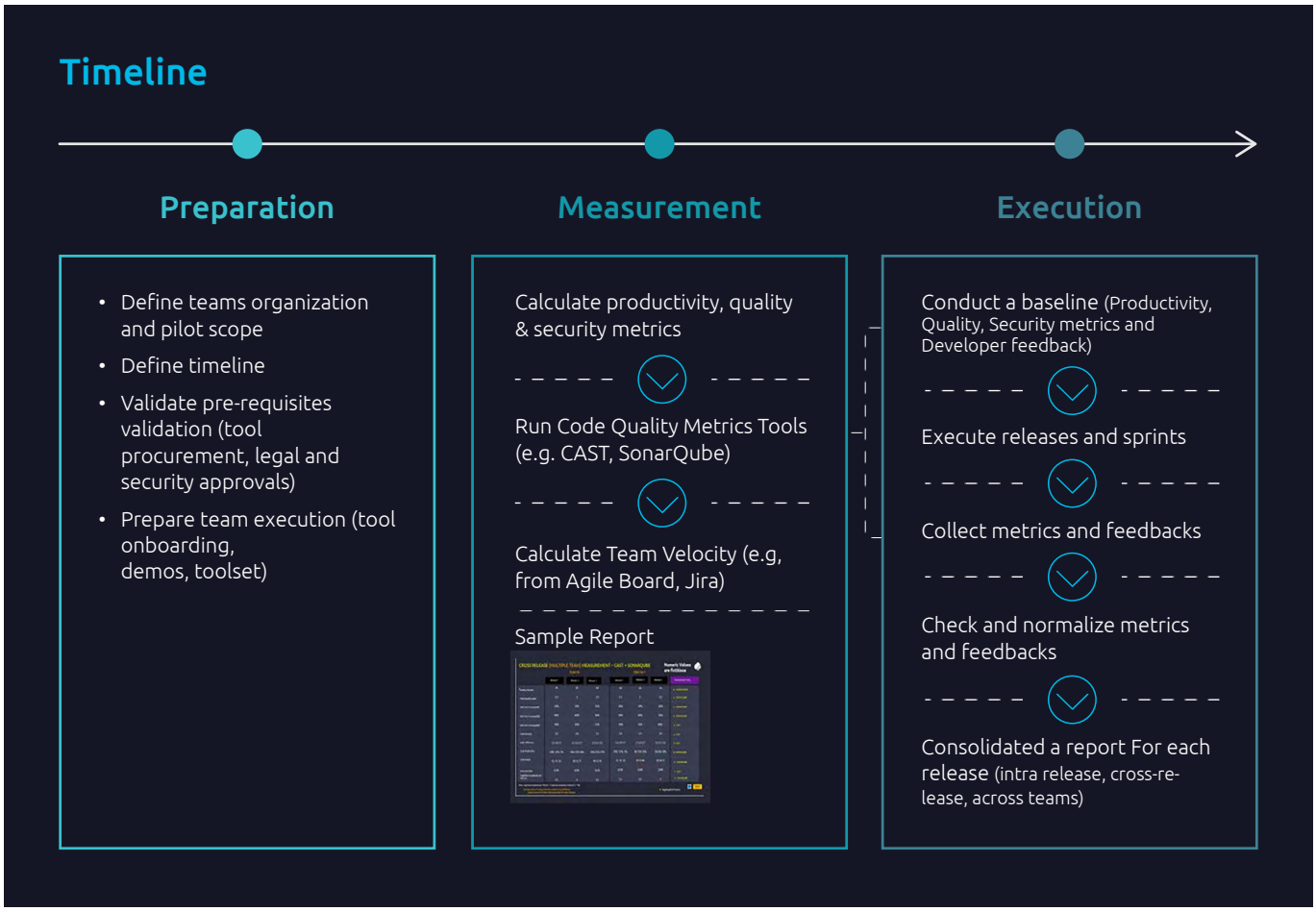
- **Senior pyramid of team members:** Highly skilled and capable team. No need for coaching, mentoring, or detailed code reviews. In an ideal world, this is the gold standard.
- **Well-balanced pyramid of team members:** Good mix of seniors and juniors. Coaching, mentoring, and reviewing are undertaken by senior team members in parallel to daily work.
- **Junior pyramid of team members:** A majority of juniors. This demands a focus on coaching, mentoring, and reviewing, as there are just a few senior team members.



Process – What are the key considerations?

Once all the components have been defined, a process is needed to ensure high quality results and to reduce side effects due to estimation inaccuracy and the human factor.

- Create the teams organization and the experimentation scope, including the use cases, the backlog, and the technology stack.
- Define the timeline and the measurement approach, including the duration, the phases, and the process.
- Validate the prerequisites and the success factors, such as the team stability, the legal clearance, the cybersecurity approval, and the tooling setup.
- Conduct a baseline audit to understand the current situation of the software engineering metrics, such as the coding velocity, the code quality, the code security, and the developer experience, without Gen AI assistance.
- Execute sprints and releases with Gen AI assistance, using the selected Gen AI tools and following the best practices and guidelines.
- Collect the metrics and the feedback during and after the real size experimentation execution, using the measurement tools and the surveys.
- Check and normalize the metrics and the feedback, using the normalization process and the formulas.
- Consolidate and report the measurement results, using the templates and the formats, and highlighting the key insights and findings.



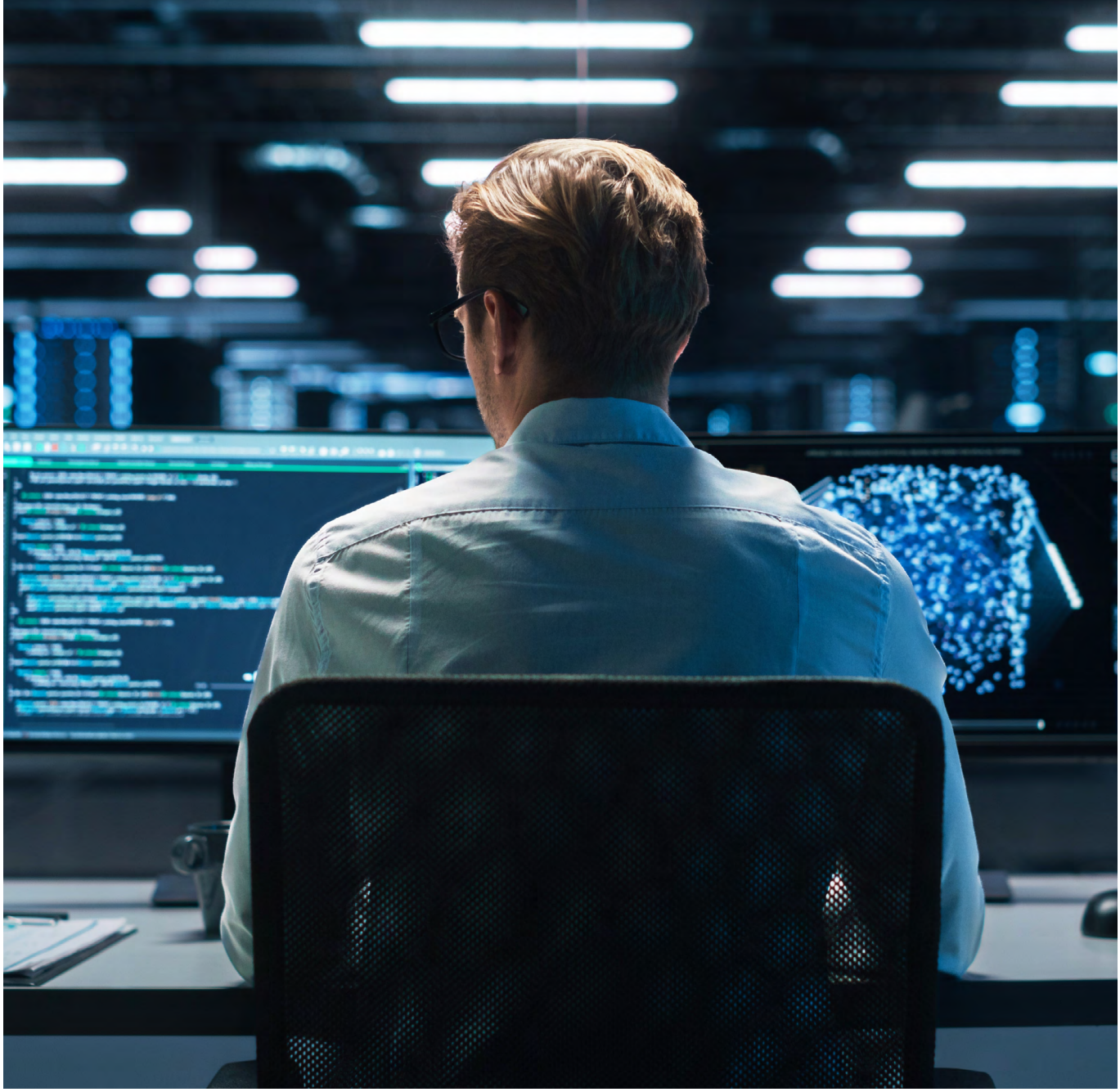
[9] *Capgemini Research Institute "Turbocharging software", June 2024*

Result: Key insights from real size experimentations with measurement

At Capgemini we ran several real size experimentations, both internally and externally together with clients. The results are positive and encouraging, as summarized here.

- **Developers love it:** The qualitative feedback from team members has been consistently positive. They love the new tools and the different way of working. As stated in our report, generative AI tools can help junior professionals learn faster and come up to speed quickly, while they allow senior professionals to focus on grooming juniors by ensuring their learning and retention, solving complex issues, and collaborating with business.
- **Velocity goes up:** Each experiment has its own context and specifics, but overall, we see a 10-30% improvement in coding and unit testing as part of the software development lifecycle.
- **Benefits for all:** Figures clearly show that junior pyramid teams benefit most, but senior pyramid teams also benefit measurably.
- **Code quality is key:** We see no degradation in code quality as measured by static code analysis and manual code reviews.
- **Documentation available:** As the 'boring' documentation task was taken over by the co-real size experimentations, the relevant metrics improved.
- **Unit test in place:** The generation of unit tests improved overall test code coverage, which is an important metric for functional correctness.

[9] *Capgemini Research Institute "Turbocharging software", June 2024*



Conclusion

Measurement:

The journey from data to decisions in the realm of Gen AI hinges on the robustness of measurement frameworks. These frameworks serve as the backbone of effective evaluation and benchmarking, enabling us to quantify the impact of AI tools accurately and consistently. By establishing clear objectives, standardizing data collection, and setting baseline metrics, we can gain deep insights into the performance and value of Gen AI in our workflows.

The defined measurement protocol is proven and helps us to better understand the effects of Gen AI in coding, and unit testing. As it's very flexible, and not bound to any specific tool, everyone can implement it in the given environment and benefit from benchmark data and a proven approach.

For more information on Capgemini's Generative AI for software engineering offer, visit:

www.capgemini.com/solutions/generative-ai-for-software-engineering/

About the authors

Pierre-Yves Glever

Executive Vice President, Global Head of C&CA,
Capgemini

pierre-yves-glever@capgemini.com

Stephane Girard

CTO, Global C&CA,
Capgemini

stephane.girard@capgemini.com

Thilo Hermann

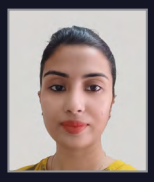
CTO Deputy, Global C&CA,
Capgemini

thilo.hermann@capgemini.com

Sripriya Venkatesan

Director and Chief Architect, Global C&CA,
Capgemini

sripriya.venkatesan@capgemini.com



Farah Naaz

Senior Consultant – UI Architect, Global C&CA,
Capgemini

farah.a.naaz@capgemini.com

Lorna Neville

Marketing Director, Global C&CA,
Capgemini

lorna.neville@capgemini.com

About Capgemini

Capgemini is a global business and technology transformation partner, helping organizations to accelerate their dual transition to a digital and sustainable world, while creating tangible impact for enterprises and society. It isa responsible and diverse group of 340,000 team members in more than 50 countries. With its strong over 55-year heritage, Capgemini is trusted by its clients to unlock the value of technology to address the entire breadth of their business needs. It delivers end-to-end services and solutions leveraging strengths from strategyand design to engineering, all fueled by its market leading capabilities in AI, cloud and data, combined with itsdeep industry expertise and partner ecosystem. The Group reported 2023 global revenues of €22.5 billion.

www.capgemini.com

Capgemini

Get the future you want