



GitHub Copilot workshop

Discover how **GitHub Copilot & Generative AI** can transform your organization

Our agenda for today

Welcome, Introductions, and Objectives

GitHub Copilot overview

Security and Copyright

Prompt Engineering

GitHub Copilot in action

Next Steps

Topics we'll cover:

1. Gen AI in the IDE (generate code, documentation, auto complete)
2. Security controls (how to prevent malicious code being introduced. How to prevent proprietary information from being disclosed.)
3. Legal copyright (how to ensure we are not using public code that could risk losing rights to the products / services we build.)

GitHub Copilot Overview

GitHub Copilot
Overview

Security and
Copyright

The Power of
Prompt
Engineering

GitHub Copilot
in Action

Next Steps

Slalom's Test of **Productivity with Copilot**

Slalom conducted a test with developers in 2 offices and saw a 20-50% productivity gain from using Copilot

Slalom's Experiment:

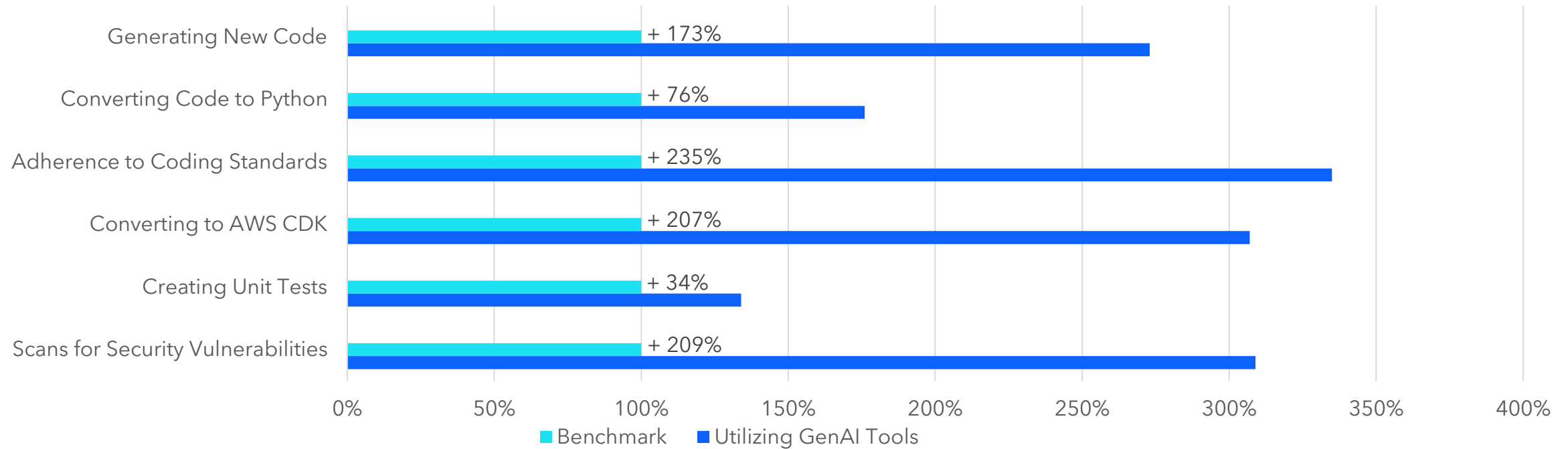
- 4 Pods, 2 in London, 2 in Denver
- Identical code bases & backlogs
- Half used GitHub Copilot, half did not
- Developers had no training or previous experience on the codebase or on GitHub Copilot
- Pods were given 1 Sprint to complete the work

What we Saw:

- **20-50% faster code production** with Copilot
- Less experienced engineers saw a bigger productivity boost
- Code Quality varied by language
- Led to more unnecessary code duplication in some languages

Massive Productivity Gains with GenAI Tooling

OBSERVED PRODUCTIVITY INCREASE BY USE CASE (# OF LINES OF CODE ADDED OR DELETED PER MINUTE)



I found it beneficial not having to shift back and forth with the browser (Google, Stack Overflow), it was able to find good answers quickly, and can read the context of the question

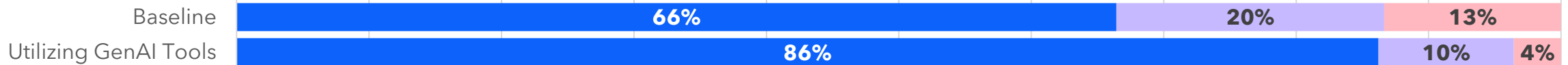
Allows developers to come up to speed faster because the code is standardized and uniform

The tools enables Developers to quickly convert between languages without requiring a deep understanding of both languages, allowing much faster time to productivity

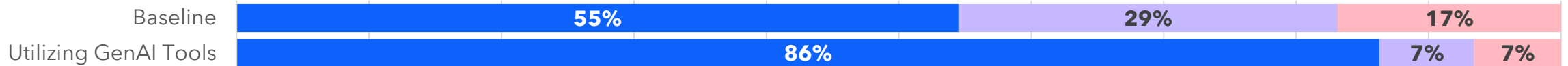
While Improving Developer Satisfaction

DEVELOPER SATISFACTION WITH KEY AREAS OF JOB BEFORE AND AFTER PILOT (AVERAGES ACROSS ALL USE CASES)

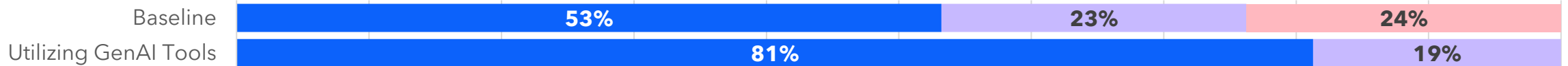
Productivity Satisfaction



Code Quality Satisfaction



Standardization Satisfaction



Flow State Satisfaction



Overall Satisfaction



■ (Strongly) Satisfied ■ Neutral ■ (Strongly) Dissatisfied

Focus on **Problem Solving**

Developers using GitHub Copilot:

Spend More Time:

- Thinking about how to structure code to solve business problems
- Writing pseudo-code and code comments of the problems they are solving
- Reviewing and tweaking AI-generated code

Spend Less Time:

- Writing boilerplate and repetitive code
- Searching google and Stack Overflow

“I can spend more time focusing on solving and writing complex business logic and less time writing unit tests”

What It's Like as a Developer

Code suggestions right in your IDE, with the context of your IDE.

```
0 references | 0 changes | 0 authors, 0 changes
39 public static void CreateTable()
40 {
41     using (var context = new TaskContext())
42     {
43         context.Database.ExecuteSqlRaw("CREATE TABLE tasks (id INT PRIMARY KEY, title VARCHAR(50), priority INT)");
44     }
45 }
46
47 // Generate a function that retrieves DB items by priority
48 public static List<Task> SelectTaskByPriority(int priority)
49 {
50     using (var context = new TaskContext())
51     {
52         var query = from task in context.Tasks
53                     where task.Priority == priority
54                     select task;
55
56         return query.ToList();
57     }
58 }
59
60
61
62
63
64
```


What is AI-Powered Code Generation?

Code Generation Powered by GitHub Copilot and OpenAI

- AI-driven suggestions for code completion
- Generate code in real time based on context and patterns
- Single, Specific and Short prompts
- Automatically draft code comments based on your code
- No need to context-switch while developing

How GitHub Copilot works

Copilot makes its suggestions based on [code available in its vector library](#), and uses your code to help with recommendations, [without storing your code in its library](#).



Security and Copyright

GitHub Copilot
Overview

**Security and
Copyright**

The Power of
Prompt
Engineering

GitHub Copilot
in Action

Next Steps

Code Exposure Outside of Slalom & Client Environments

While this is a common concern for all Gen AI technologies, GitHub Copilot has it covered!

GitHub Copilot transmits snippets of your code from your IDE to the GitHub Copilot backend in the form of a prompt to provide suggestions to you.

[Code snippets](#) data is only transmitted in real-time to return suggestions and is [discarded once suggestions are returned](#).

Copilot for Business does NOT retain any code snippets data.

Copilot for Business does NOT use your code as training data for the underlying Large Language Model

Reference: [GitHub Copilot Privacy Policy](#)

Safety of Executing Generated Code

Code is generated based on open source in GitHub and available elsewhere on the web, using your IDE's context to make suggestions specific. **There is no guarantee that it works or is safe**



Copilot Doesn't Replace Developers

- Like with code found on Stack Overflow, you shouldn't run it if you don't understand it
- Developers need to understand their code, Copilot is just here to help them be more productive and learn more quickly



Testing & Reviews are a Necessity

- Slalom Development Practices should still be followed: All code is unit tested and QE'ed, and peer-reviewed before merged
- All Slalom Copilot training emphasizes this as a core requirement to usage!

Filtering Mechanism to **Mitigate Risk**

GitHub Copilot includes an optional code referencing filter to detect and suppress certain suggestions that match public code on GitHub.

- GitHub has created a duplication detection filter to detect and suppress suggestions that contain code segments over a certain length that match public code on GitHub. This filter can be enabled by the administrator for your enterprise and it can apply for all organizations within your enterprise, or the administrator can defer control to individual organizations.
- With the filter enabled, Copilot checks code suggestions for matches or near-matches against public code on GitHub of 65 lexemes or more (on average, 150 characters). If there is a match, the suggestion will not be shown to the user.

In addition to off-topic, harmful, and offensive output filters, GitHub Copilot also scans the outputs for vulnerable code

Copilot for Business: Security and Privacy

GitHub Copilot for Business is an AI pair-programmer that helps developers write code faster with less work. GitHub Copilot is powered by GitHub customized variants of OpenAI's models, which are generative pretrained Large Language Models optimized for code assistance.

GitHub is committed to transparency around privacy and security of all their products.

Copilot for Business:

- Does NOT retain user content (e.g. code from user's editor)
- Does NOT use your code to Improve the model
- Does NOT retain suggestions produced by Copilot
- Does NOT send information to OpenAI, the software company
- Is governed by both GitHub Enterprise and Copilot's product specific terms

GitHub as Data Controller

Subject to authorization through GitHub's Data Protection Agreement (DPA), GitHub processes UED as a data controller only for following purposes:

- Billing and account management
- Compensation, such as calculating employee commissions and partner incentives
- Aggregated internal reporting and business modeling, such as forecasting, revenue, capacity planning, and product strategy
- Aggregated financial reporting

For both its controller and processor roles, GitHub applies the principles of data minimization.

Sources of Suggestions

GitHub Copilot is powered by OpenAI Large Language Models (LLMs). They are trained on natural language text and source code from publicly available sources. OpenAI does not disclose its training set or specify all of its sources, however, the models are not trained on any code from private repositories.

GitHub Copilot does not "copy and paste" code used to train its model - rather, it generates new code in a probabilistic way, and the probability that a suggestion contains the same code as a snippet utilized in training is low. Previous research showed that many of these cases happen when GitHub Copilot is unable to glean sufficient context from the code you are writing, or when there is a common, perhaps even universal, solution to the problem.

GitHub Copilot also checks whether code suggestions match code in GitHub public repositories. Any suggestions matching at 150 characters or more (excluding whitespace) will be flagged. Organization owners and admins can elect to have such matches blocked before they are shown.

GitHub as Data Controller

Subject to authorization through GitHub's Data Protection Agreement (DPA), GitHub processes UED as a data controller only for following purposes:

- Billing and account management
- Compensation, such as calculating employee commissions and partner incentives
- Aggregated internal reporting and business modeling, such as forecasting, revenue, capacity planning, and product strategy
- Aggregated financial reporting

For both its controller and processor roles, GitHub applies the principles of data minimization.

Sources of Suggestions

GitHub Copilot is powered by OpenAI Large Language Models (LLMs). They are trained on natural language text and source code from publicly available sources. OpenAI does not disclose its training set or specify all of its sources, however, the models are not trained on any code from private repositories.

GitHub Copilot does not "copy and paste" code used to train its model - rather, it generates new code in a probabilistic way, and the probability that a suggestion contains the same code as a snippet utilized in training is low. [Previous research](#) showed that many of these cases happen when GitHub Copilot is unable to glean sufficient context from the code you are writing, or when there is a common, perhaps even universal, solution to the problem.

GitHub Copilot also checks whether code suggestions match code in GitHub public repositories. Any suggestions matching at 150 characters or more (excluding whitespace) will be flagged. Organization owners and admins can elect to have such matches blocked before they are shown.

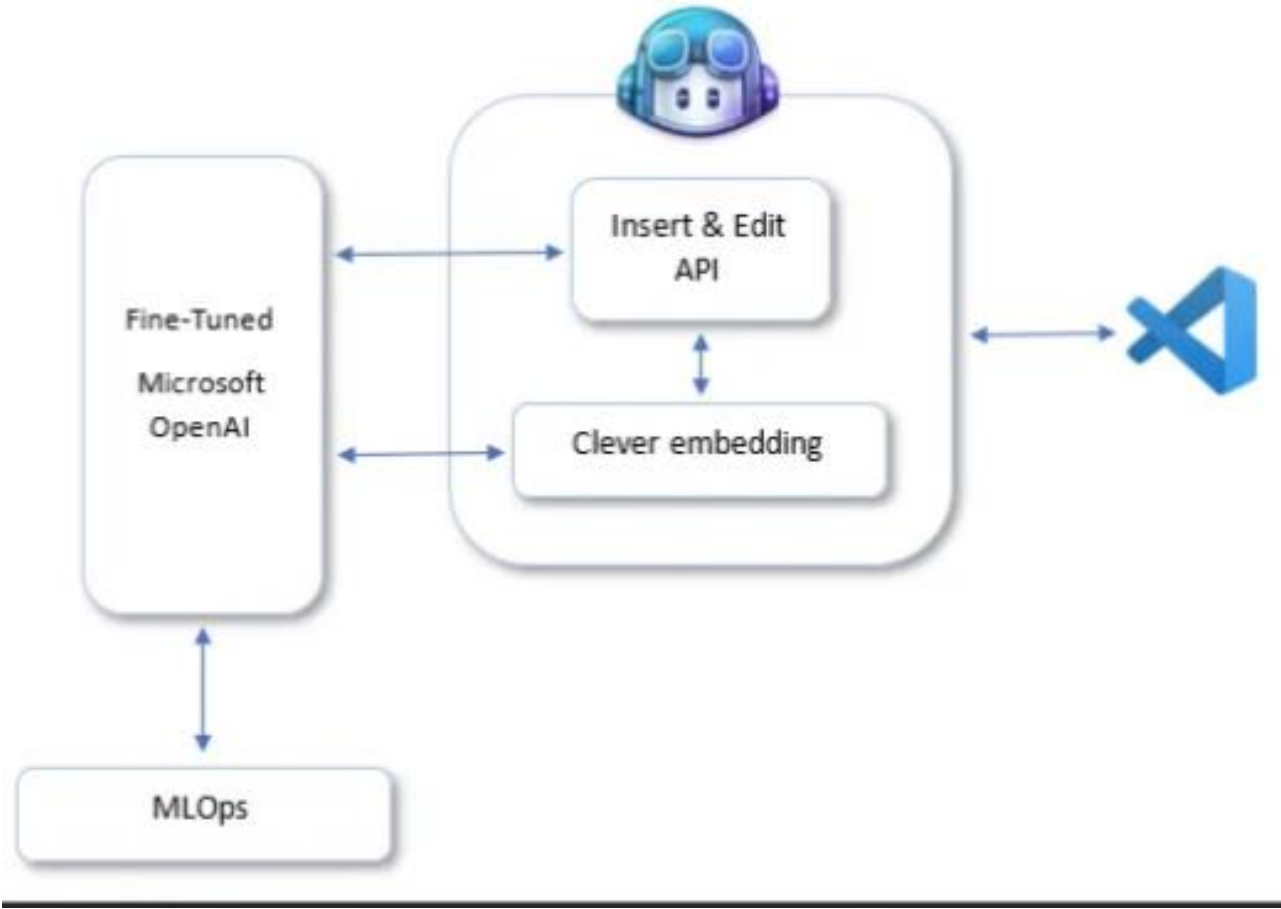
Copilot For Business: Security and Privacy

GitHub Copilot for Business is an AI pair-programmer that helps developers write code faster with less work. GitHub Copilot is powered by GitHub customized variants of OpenAI's models, which are generative pretrained Large Language Models optimized for code assistance. While GitHub Copilot is groundbreaking technology, GitHub recognizes the need for enterprises to have an understanding of the security behind the tools they choose to drive their business. GitHub is committed to transparency around privacy and security of all our products. As GitHub Copilot for Business is incorporated into our compliance reporting in the coming months, this white paper is meant to provide the necessary transparency for Copilot for Business privacy and security practices.

Copilot for Business:

- Does NOT retain user content (e.g. code from the user's editor)
- Does NOT use your code to improve the model
- Does NOT retain suggestions produced by Copilot
- Does NOT send information to OpenAI, the software company
- Is governed by both GitHub Enterprise and Copilot's [product-specific terms](#)

This document is for the GitHub Copilot for Business extension **only**.



The Power of Prompt Engineering

GitHub Copilot
Overview

Security and
Copyright

The Power of
Prompt
Engineering

GitHub Copilot
in Action

Next Steps

The “Psychology” of Large Language Models

- They don’t “think” then write, they only write
- They don’t edit their answer or correct their mistakes
- They don’t know what they don’t know
- They don’t understand their own capabilities
- They don’t reflect on their answer
- **They predict the next token (word)**

Prompt Engineering can [help to make up for those limitations](#) and make the model generate high quality outputs.



Prompt Engineering Principles for Crafting & Creating Effective Prompts



Know your model

- Different language models have different capabilities, strengths and weaknesses
- Some models are not trained on code, not trained to perform well with complicated reasoning, or not trained on recent data
- This is the most important principle of crafting effective prompts



Be detailed and specific

- Provide detailed information about your desired output using direct language
- Focus the prompt on a specific problem, topic, question or output type. Multiple asks in a single prompt can make it difficult for a model to answer effectively



Provide context

- Contextual information helps the model to solve the task and generate more appropriate responses
- Include conversation history, examples of desired output, or the model's role in the prompt



Experiment

- Try different approaches to prompting. Crafting effective prompts is more of an art than a science
- Alter prompt wording, change the prompt to align with the model's capabilities, and even ask the model for help to optimize the prompt

GitHub Copilot demo

on-demand AI Pair Programmer

GitHub Copilot
Overview

Security and
Copyright

The Power of
Prompt
Engineering

GitHub Copilot
in Action

Next Steps

Objectives



Demonstrate how GitHub Copilot speeds up the development process and in what specific areas it provides the most value.



Provide insights about what GitHub Copilot does - and does not do - through best practice usage considerations.



Understanding the recommended deployment approach for GitHub Copilot in your company factoring in comms & team engagement.

GitHub Copilot Chat

GitHub Copilot
Overview

Security and
Copyright

The Power of
Prompt
Engineering

GitHub Copilot
in Action

Next Steps

Points to remember

- Developer is still responsible for the coding and for navigating the copilot in the right direction.
- Just like a pair programmer needs some context, copilot also needs context.
- It is available on demand unlike real programmer for pairing
- Remembers all the syntax. Trained by almost all the programming languages available in public repository
- Quality is still developer's responsibility. Copilot will help with it.

Next Steps

GitHub Copilot
Overview

Security and
Copyright

The Power of
Prompt
Engineering

GitHub Copilot
in Action

Next Steps

How we're using **AI to accelerate product engineering**

Backlog refinement and planning

- Automate backlog management, prioritizing items based on complexity and business value
- Predict story points for new user stories using historical data
- Enhance sprint planning with an optimal mix of user stories for each sprint based on complexity and team velocity

Design and develop

- Provide large blocks of code from natural language requests
- Add to code written through copilot functionality
- Analyze and understand technical debt, refactor and simplify complex solutions, and generate code and templates

Testing

- Generate unit and integration tests
- Identify and create code to test security vulnerabilities
- Generate synthetic data for data platform modernization programs
- Automate code reviews
- Predict potential system failures or bugs
- Create rapid prototypes

GitHub Copilot Implementation Approach

Key Considerations for your adoption plan:

1. **Manage like a project**

Define your champions and team to realise the value of the tool

2. **Plan your phasing**

Be deliberate in your team choice for usage

3. **Deliver an outcome**

Use this as an opportunity to address a pain point / new feature

4. **It's all about engineering culture**

Promote it, critique it, show the value to the team!

Next Steps

Post-Workshop Offerings









Take next steps in realizing the value that AI can bring

Our one-day workshops are designed to ready you for next steps in realizing the value that AI can bring. Working together with you, we will identify which of the following options will be a most effective follow-on for bringing the value of AI into your engineering organization. Please note the following are recommendations based on prior experience. Slalom will customize each offering for RCCL.

Business Case & Pilot	Team Training	Organization Rollout
<p>With this option Slalom will work with you to identify and prove out the business case for bringing GitHub Copilot into your engineering organization.</p> <p>What it Looks Like</p> <ul style="list-style-type: none">• ~6 weeks timeline• 10-50 participants• ~15-20 lab hour sessions held <p>Output/Outcome</p> <ul style="list-style-type: none">• Success metrics identified, tested, and documented• Participants trained and supported during pilot• Business case with ROI created• Wider rollout and adoption plan created	<p>Slalom works with your organization to develop and deliver training to a group of your engineers.</p> <p>What it Looks Like</p> <ul style="list-style-type: none">• ~6 weeks timeline• 25-100 engineers trained and supported• 1 week planning and alignment -> 2 development sprints -> 1 week closure and wider adoption planning <p>Output/Outcome</p> <ul style="list-style-type: none">• Customized team training created and delivered• Knowledge Base created and updated• Team members trained as go-forward trainers• Wider rollout and adoption plan created	<p>Work together on a program-scale initiative that brings GitHub Copilot to transform your organization.</p> <p>What it Looks Like</p> <ul style="list-style-type: none">• 12-25 weeks• Initial boarding groups, rolling out to• 4 weeks planning -> 8 weeks (4 development sprints) activation -> 2 weeks wider rollout/scaling <p>Output/Outcome</p> <ul style="list-style-type: none">• Success metrics identified, tracked, and reported on throughout• Customized trainings created, Train-the-trainer content developed, trainers trained• Trainings, office hours, and support provided by Slalom and/or your experts with Slalom'• Rollout plan documented and executed

Potential Measures of Development Success

Type	Metric	Measurement Approach	Expected Impact
Quantitative	Speed to Market	Comparison pre- and post-pilot Metrics should increase over time as team adapts	
	Sprint Velocity		
	Code Commits per sprint		
	Number of Bugs	Comparison pre- and post-pilot Metrics should not deteriorate	
	Code Coverage		
	Code Complexity		
Qualitative	Developer Satisfaction	Retrospectives and people management	Reduction in time spent doing mundane tasks
	Developer Engagement	Team surveys / pulse checks for engagement	Energy conserved by spending more time "in the flow"

GitHub Copilot Rollout Plan (Example Only)

Initial Focus - Intake Team (5 Developers OPAD Team) for 2 Sprints

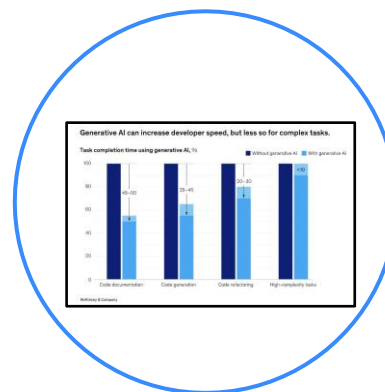
Workstream	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6
Mobilisation	Agree Pilot Team & Scope					Wider Adoption Plan
	IDE Setup					
Pilot Training and Awareness	Training - Usage Guidelines					Train the Trainer Approach
	Best Practices KB	Ongoing KB Updates	Ongoing KB Updates	Ongoing KB Updates	Ongoing KB Updates	
Pilot Delivery		Sprint 1	Sprint 1	Sprint 2	Sprint 2	
Metrics & Impact Measurement	Metric Agreement	Reporting Setup	Sprint 1 Metrics Review		Sprint 2 Metrics Review	
			Team Checkpoint		Team Checkpoint	
Tech LT Engagement	Pilot Sign Off		Pilot Checkpoint Report			Pilot Closure Report
Team Culture	Establish Team Chat for Q&A	Standup Check-ins on Copilot	Standup Check-ins on Copilot	Standup Check-ins on Copilot	Standup Check-ins on Copilot	
	Team Comms on Pilot		Spot Prize - KB Submission		Spot Prize - KB Submission	

Research confirms that Copilot Speeds-up “Dev Task Completion”



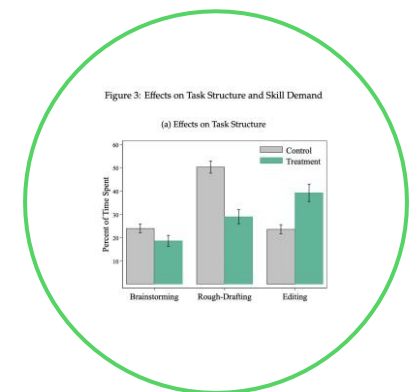
GitHub 2022

95 Devs, Same Task
45 Used Copilot
55% Faster
10% more likely to complete



McKinsey

45-50% Faster for Code Docs
35-45% Faster for Code Gen
20-30% Faster for Refactoring
<10% for Complex Coding Tasks



MIT

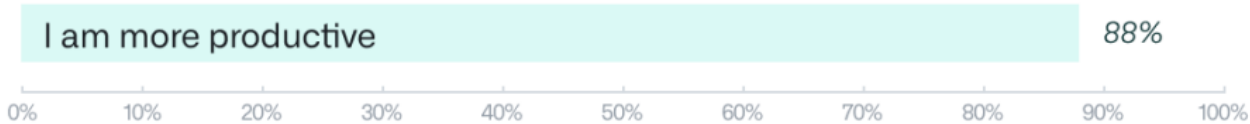
Changes the Structure of Tasks:
Brainstorming 25%
Writing Rough Draft 50% -> 25%
Editing 25%-> 50%
Reduces inequality

Microsoft's Study Also Found Significant Productivity Gains

Microsoft conducted a [survey](#) of more than 2,000 users of the Copilot technical preview, and found:

When using GitHub Copilot...

Perceived Productivity



Efficiency and Flow*



Helping Developers **Learn and Grow**

Aids in faster learning of new languages and frameworks



Experienced Developers in a New Space

Copilot speeds up the process of learning how to build in a new framework



Newer Developers Upskilling

Slalom's tests showed larger productivity gains from using Copilot for more junior developers

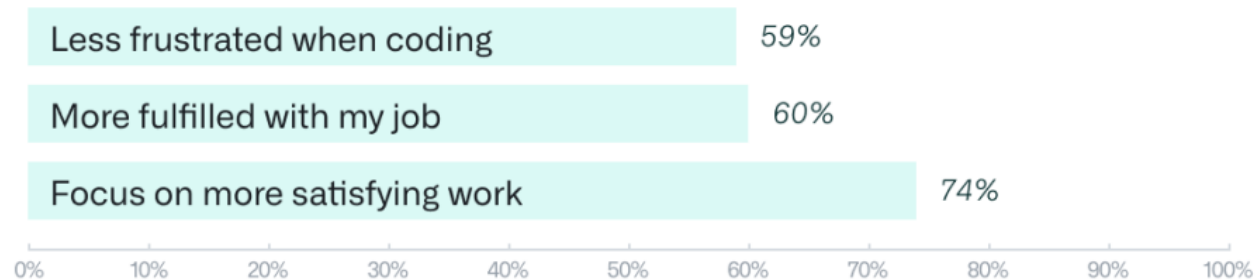
"Copilot makes it faster and easier for me to learn how to build Mobile apps in React Native"

Focus on Problem Solving

Microsoft conducted a [survey](#) of more than 2,000 users of the Copilot technical preview, and found:

When using GitHub Copilot...

Satisfaction and Well-being*



"I'm able to do the simple things much more quickly, leaving more time for the hard problems"

It Helps Human Developers, it Doesn't Replace Them

The Code Isn't Perfect

- While some code suggestions work perfectly, many need to be tweaked or downright ignored
- While Copilot uses the context of what's open in your IDE, it doesn't (yet) use the full context of your project

Doesn't Have Business Context

- It doesn't know the business problem you're trying to solve - its recommendations are based on your code & comments
- It doesn't read your Jira backlog, requirements documents, or talk to your solution owner - it just guesses based on your IDE and what you're working on

"Three times this week I said: oh my god I can't believe how good this is. It's so intelligent, I love it"

It still has **limitations**

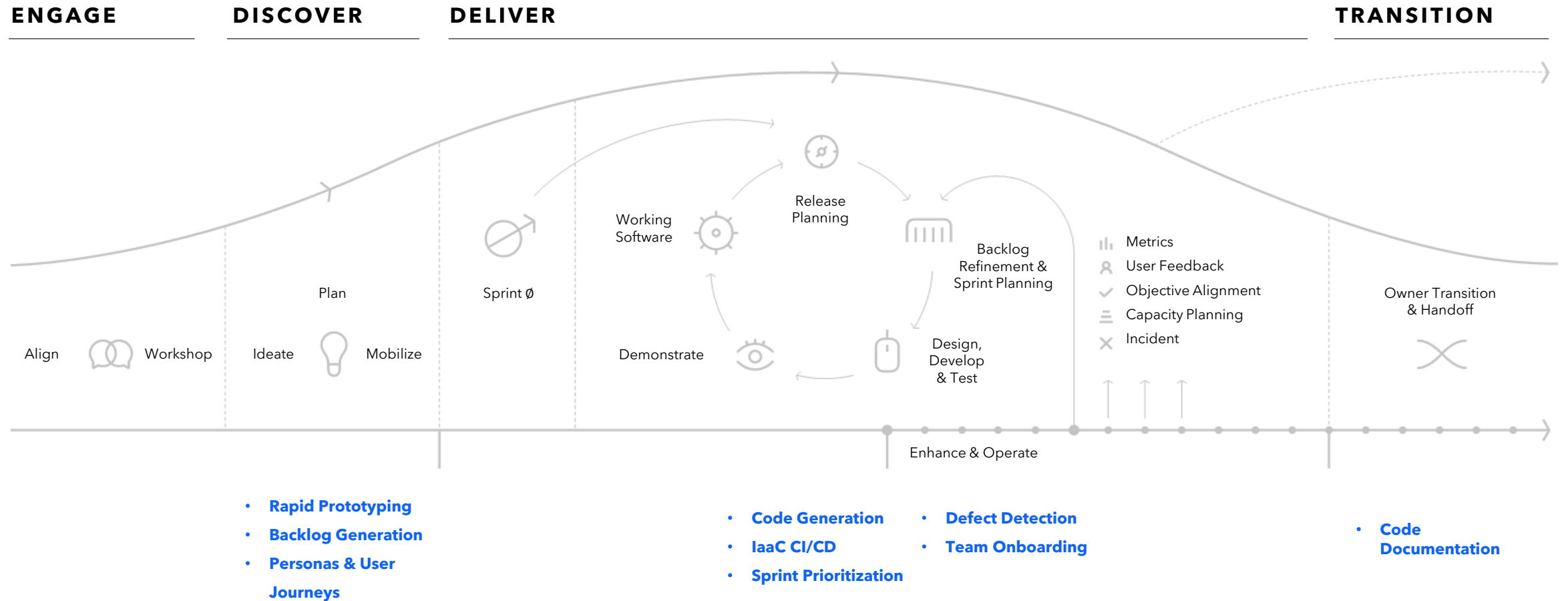
What we see, so far:

- Continues to be mind-blowing, even as we learn its limitations
- Earlier-career engineers see a bigger boost
- Different quality language-by-language
- Context is limited to code, humans must bring in the application and outside understanding (but this is improving) – It's not "Your AI pair programmer"
- Models are trained to a point in time—it struggles with recently updated frameworks
- Can introduce bugs
- Can encourage bad habits
- Not "safe" in the way refactoring tools, such as ReSharper, are

"While it's not always perfect, I can't believe how it speeds up my work and lets me focus on the fun problem-solving"

Gen AI & Software Development

We are piloting Gen AI at all stages of Slalom's PEM (*Product Engineering Methodology*) to reduce time to market and increase deployment frequency



Thank you

Appendix – Typical POC



Indicative POC approach

- An open invitation to [collaborate](#) on a Slalom supported POC
- Microsoft strategic [investment](#) supports available
- You bring [a](#)) Senior Stakeholder Sponsorship [b](#)) your knowledge and [c](#)) your availability to support
- We bring [a](#)) The Gen AI platform [b](#)) the methodology [c](#)) the team to build it **with you**

How

We begin by aligning on objectives and outcomes for the solution - and move quickly and iteratively to requirements and design.

We bring your vision to life by developing a tangible, interactive simulation of an idea that can be used to demonstrate and validate the art of the possible, while concurrently building the long-term vision and business case.

What

We partner with you to:

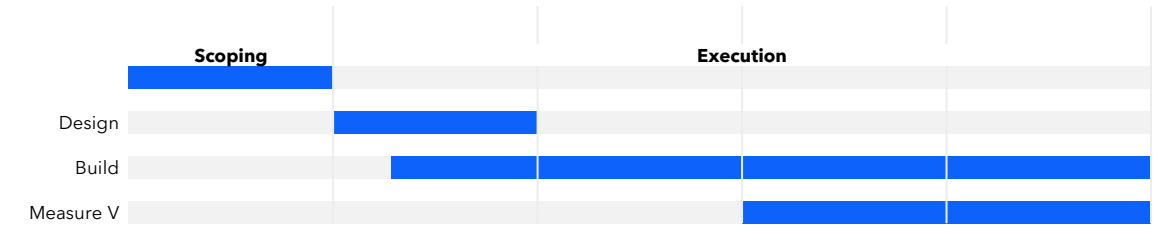
- Align on the business objectives and desired outcomes
- Define the requirements and develop a backlog
- Define scope of the POC
- Design and build the POC
- Define the future vision of the solution and develop a business case

Outcome

At the end of this engagement, you will have a functional proof of concept, as well as documented product recommendations, future solution considerations and next steps (including an SOW to progress the POC to production, if relevant).

You will also have a public credential to share and showcase your achievements in the generative AI space.

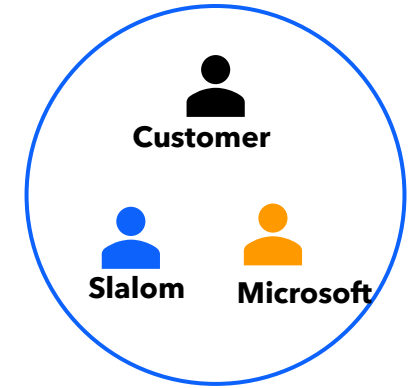
Phases and activities



SCOPING	DESIGN	BUILD	MEASURE VALUE
<ul style="list-style-type: none"> •Define business objectives and desired outcomes •Gain alignment with key stakeholders •Define POC delivery roadmap •Assess the current state •Identify, mobilize team including kicking off technical set up tasks 	<ul style="list-style-type: none"> •Create and iterate on visual representation of POC •Define business and technical requirements and build backlog •Set up technical environment and prep data 	<ul style="list-style-type: none"> •Develop POC and iterate •Collect and action insights from user testing 	<ul style="list-style-type: none"> •Develop product recommendations •Capture future solution considerations •Determine next steps

Indicative POC teaming

Microsoft and Slalom will work with you once the use case definition is clear to define the optimal team to deliver the POC.



CORE TEAM (TYPICALLY 4 - 6)

Role	Typical activities	Source
Product Owner ¹	Driving POC vision, scope and outcomes	Customer
Engineering Lead	Leading the solution design and technical implementation of the POC	Slalom
Engineer	Executing the technical delivery	Customer, Slalom
Solution Owner	Scoping and sprint execution support, driving capture of future solution considerations	Slalom

Notes:

1. Likely to be a single person, though the Product Owner may represent a larger group of business specialists. A 50-60% time allocation is required from the product owner during the execution phase.
2. Not fully dedicated to the POC delivery. Kept informed and consulted, as required.
3. There will be a single Engineering Lead nominated as part of the core delivery team, they will have access to the full range of Slalom engineering expertise.

WIDER ENGAGEMENT SUPPORT TEAM²

Role	Typical activities	Source
Project Sponsor	Providing resources, approvals, decisions and owning the vision, POC and desired outcomes.	Customer
Business Domain Specialist ¹	Providing process knowledge and business expertise	Customer
Data Specialist	Providing understanding of the data, supporting translation of the business needs into technical outcomes	Customer
Engagement Lead	Providing Microsoft governance and support, and issuing project communications	Microsoft
Microsoft Account Team	Providing Microsoft technical expertise, guidance and support	Microsoft
Engineering Leads ³	Solution approach guidance in specialized field e.g., platform engineering, quality engineering, data engineering, software engineering	Slalom
Experience Architect	Providing guidance from a user experience perspective on the POC and/or future solution	Slalom