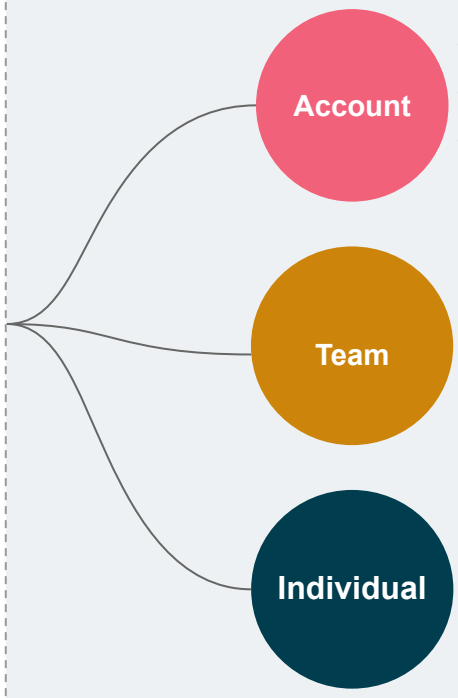
# Problem statement for launching an AI Capability Growth

## Client expectation

- Deploy OOTB (Out of the Box) AI enablement capabilities internally to drive efficiency and effectiveness across all of organization.
- There is absolutely no excuse for not making AI assisted development part of your workflow. With all the productivity evidence and free assistance and training provided, **I would not even consider engineers for a job without skills and experience using AI dev tools.** Invest time in your career.

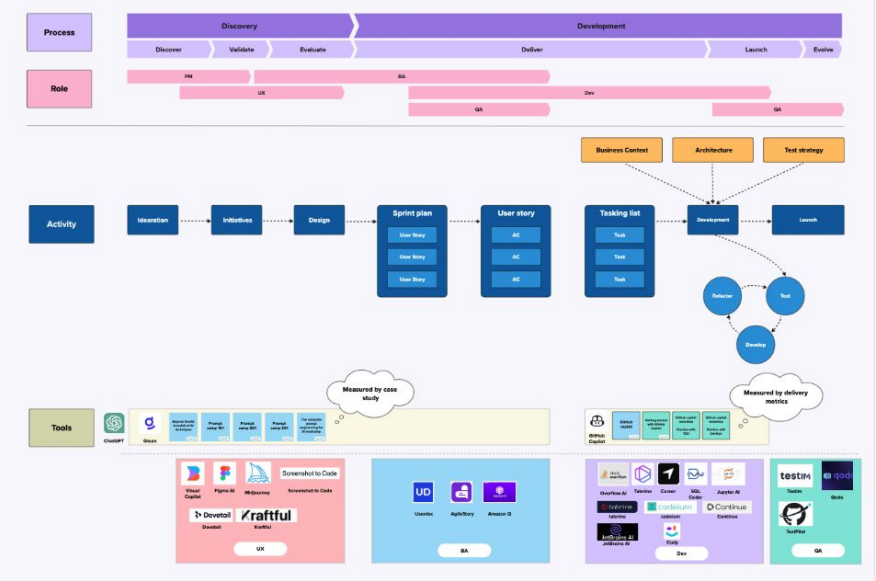
## Industry Trend

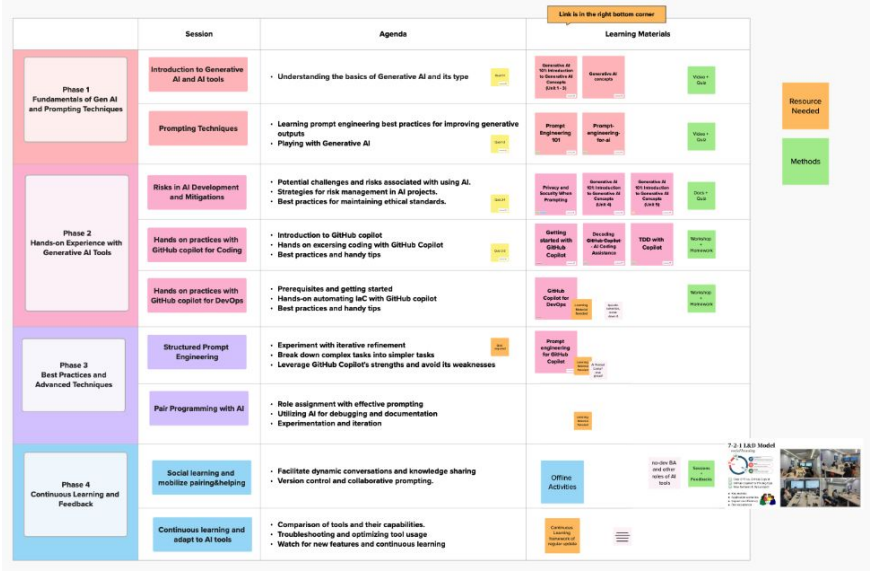
- We believe there is an opportunity for TWs delivery teams to leverage AI tooling to help us deliver more efficiently to assist our clients in getting value to market faster.




- #1: No clear and achievable expectations for the use of AI tools
- #2: No clear guidance on how to measure capability growth
- #3: No clear measure of success from the use of AI
- #4: Not yet able to hands on development
- #5: Not sure how to achieve best practices in a team context
- #6: AI Development is progressing very rapidly and it can be confusing for individual to figure out what to use when
- #7: Hard to follow the current trends

# AI Capability Growth Framework




Building AI Capability Growth framework based on Discovery-and-Delivery-Process to solve #1, #3, #4, #5, #7. It allows us to look at the integration of AI in each activity from an agile engineering perspective, and the potential tooling options also allow us to keep our eyes on development trends.

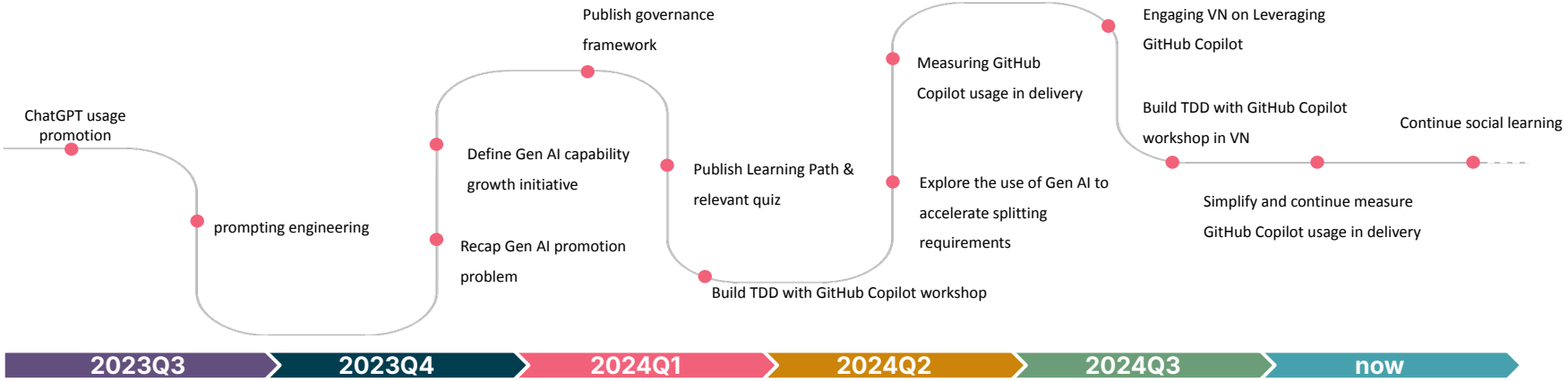
From a practical perspective, a learning path map is developed to enable individuals to learn the fundamentals, hands-on practice and trendy development directions end-to-end to address #2, #5, #6, #7

AI Capability Growth Roadmap

Link is in the right bottom corner

| | Session | Agenda | Learning Materials |
|---|---|---|---|
| Phase 1<br>Fundamentals of AI and Language Models | Introduction to Artificial Intelligence | • Overview of AI, machine learning, and deep learning.<br>• Differences between narrow AI and general AI.<br>• Fundamentals of Language Models.<br>• Introduction to Natural Language Processing (NLP). | An executive's guide to AI<br>Splunk AI & ML<br>Machine Learning Essentials (Section 1)<br>Overview of NLP (Section 3 - 4) |
| | Use Cases of AI in Software Development | • Exploration of AI in the software development lifecycle.<br>• Real-world examples of AI implementation in software projects. | LLM empowers R&D efficiency improvement<br>AI-Assisted Software Delivery<br>AI-assisted software development practices guideline |
| Phase 2<br>Gen AI Data and Model Understanding | Generative AI and Data Model | • Basic principles and mechanisms<br>• Model outputs and informed decision-making.<br>• Problems of biases and inaccuracies. | Generative AI 101: Introduction to Generative AI Concepts (Unit 1 - 3)<br>ChatGPT and the model behind<br>Generative AI: Learn about the next AI frontier (Section 1 - 3)<br>How chatGPT actually works |
| Phase 3<br>Risks, Ethics, and Mitigations | Risks in AI Development and Mitigations | • Potential challenges and risks associated with using AI.<br>• Strategies for risk management in AI projects.<br>• Best practices for maintaining ethical standards. | Generative AI 101: Introduction to Generative AI Concepts (Unit 4)<br>Generative AI 101: Introduction to Generative AI Concepts (Unit 5)<br>SecConf2023: Navigating risks of AI language model<br>Privacy and Security When Prompting |
| Phase 4<br>Effective Prompting Techniques | Effective Prompting Techniques | • Comparison of tools and their capabilities.<br>• Crafting prompts to get desired outputs.<br>• Tips for refining prompts for better results.<br>• Troubleshooting and optimizing tool usage. | Prompt camp 101<br>Prompt camp 201<br>Prompt camp 301<br>Beyond GenAI: A toolkit of AI techniques<br>Prompt Frameworks<br>The complete prompt engineering for AI bootcamp<br>Awesome prompting |
| | Collaborative Development with Using Gen AI | • Team collaboration strategies when using AI tools.<br>• Version control and collaborative prompting. | GitHub copilot<br>MS copilot<br>AWS CodeWhisperer<br>OpenAI codex |
| Phase 5<br>Hands-on Experience with Generative AI Tools | GitHub Copilot (Dev Only) | • Introduction to GitHub copilot<br>• Best practices and handy tips | Getting started with GitHub Copilot<br>Decoding GitHub Copilot - AI Coding Assistance |
| | Hands-on with GitHub Copilot (Dev Only) | • Prerequisites and getting started<br>• Hands-on coding with GitHub copilot | GitHub Copilot for Coding<br>GitHub Copilot for DevOps |
| Phase 6<br>Advanced Techniques and Integration | Fine-Tuning Models for Specific Domains | • Tailored models for domain-specific requirements.<br>• Hands-on experience with fine-tuning. | LLM fine-tuning on OpenAI<br>GenAI Implementation for APP Translation |
| | Cloud Integration | • Strategies for seamless integration of AI with cloud service.<br>• Automation and optimization techniques. | MS Copilot for Azure |
| | Future Trends of Gen AI Plus Software Development | • Exploration of emerging technologies and their impact on software development.<br>• Prediction on the future trajectory of Gen AI in the technology field. | GenAI: Possibilities vs. Costs<br>Generative AI Solution Exploration: PE, RAG, FT<br>Rapid LLM App Prototyping: A No Code Approach with Dify.ai |

# Our Journey of AI Capability Growth Initiative




**> 90%**

Finish 7 sets of quizzes for Learning path stage 1~4

**80.5%(Xian)**
**61.9% (VN)**

devs completed the TDD with Github Copilot workshop

**100%**

- Everyone who participates in the workshop submit at least one insight, including ticket, commit, etc
- Keep collecting insights from new member by team iteratively collecting GitHub Copilot usage data

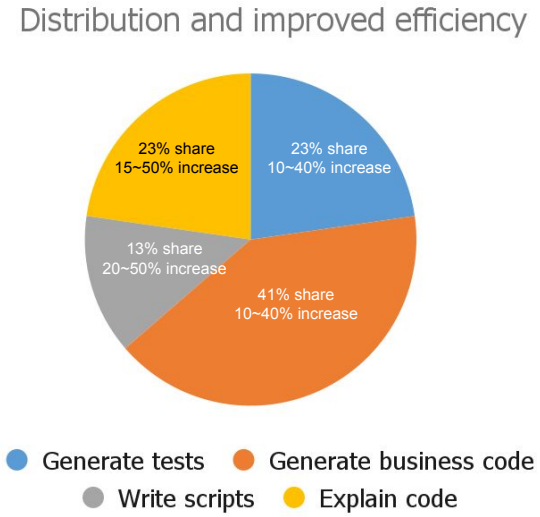
**15.8%**

Using GitHub Copilot reduced coding time for four major usage scenarios, reducing the average cycle time of each team by 15.8%

# GitHub Copilot Applicability Analytics

*Getting feedback from all team members that participated in the TDD with GitHub Copilot workshop. Picking up 1 of sprint data, these teams completed a total of 150 tickets in the recent iteration. Among them, 73 tickets did use GitHub Copilot (48.7%), 77 tickets did not use GitHub Copilot (51.3%).*

**The teams' usage scenarios can be roughly divided into four categories:**




- Generate test: including generating test code and test data. Because the code structure is simple, the accuracy is higher.
- Generate business code: the accuracy is limited by the business context, and the probability of the generated code being adjusted is relatively high, about 30% ~ 80%.
- Write scripts: Since the generated scripts are less relevant to the business, the prompting needs to contain less information and the generated code is more accurate.
- Explain code: Teams use code explanation as a more efficient way to collect business code and information, and some teams even use it to share business context in a session, especially for the legacy code.

**Scenarios where GitHub Copilot is Not used:**

- Local environment setup
- Manual tests
- Knowledge sharing
- Straightforward tasks
- **Tools decommission or upgrade**
- Designing and Solutioning
- **Alerting & monitoring**
- Supporting requests
- **Vulnerability fixing**
- Troubleshooting cross the system

# GitHub Copilot Increases Speed and Productivity
## Our Thoughtworks Teams

| Scenarios | Development time | Generate tests | | Generate business code | | Write scripts | | Explain code | | Applicable weighted cycle time saved **[S]** | General weighted cycle time saved **48.7%** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 23% **[$Z_i$]** | | 41% | | 13% | | 23% | | | |
| | | Coding time saved | Cycle time saved | Coding time saved | Cycle time saved | Coding time saved | Cycle time saved | Coding time saved | Cycle time saved | | |
| **Very optimistic** | 76.5% **[X]** | 30.0% **[$Y_i$]** | 23.0% | 50.0% | 38.3% | 50.0% | 38.3% | 40.0% | 30.6% | **33.0%** | **16.1%** |
| **Middle** | 55.4% | 28.0% | 15.5% | 29.4% | 16.3% | 38.7% | 21.4% | 22.0% | 12.2% | **15.8%** | **7.7%** |
| **Pessimistic** | 48.0% | 15.0% | 7.2% | 10.0% | 4.8% | 33.3% | 16.0% | 10.0% | 4.8% | **6.8%** | **3.3%** |

$$S = \sum_{i=1}^{n} X \times Y_i \times Z_i \quad n \in \{1, 2, 3, 4\}$$

**48.7%:** According to the most recent iteration, the team has completed about 150 tickets in total. Of these, 73 tickets did use GitHub Copilot (48.7%).

**X** = Development Time *(the percentage of time the team spends on implementation and testing during a sprint. In the table above, this value represents the average for the Xi'an team in each segment over the past two months)*

**Y** = Coding Time saved *(the percentage of coding time saved based on the analysis of each ticket collected by the team capability champion)*

**Z** = Usage Scenario Proportion *(a statistical value representing the proportion of tickets for this type of usage scenario in the sprint)*

**S** = Applicable Cycle Time Saved *(the average improvement rate of overall cycle time across teams after weighting each usage scenario (currently 4 usage scenarios))*