

PIONEERING WITH AI DRIVEN AGILE DELIVERY

BY MELON GUAN

Agile Tour
Kuala Lumpur
2023

Meet
Speaker

OCT
24



Melon Guan

Head of Delivery
#1 Bank in Hong Kong

Massive Breakthrough,
Cost Down, and Speed
Up: Pioneering with AI-
driven Agile Delivery



In today's global banking landscape, I face challenges like never before – large-scale programs, diverse teams across different time zones, and varied practices. But at Global Bank, we've been pioneering AI-driven Agile Delivery, and I'm excited to share our incredible journey with you!

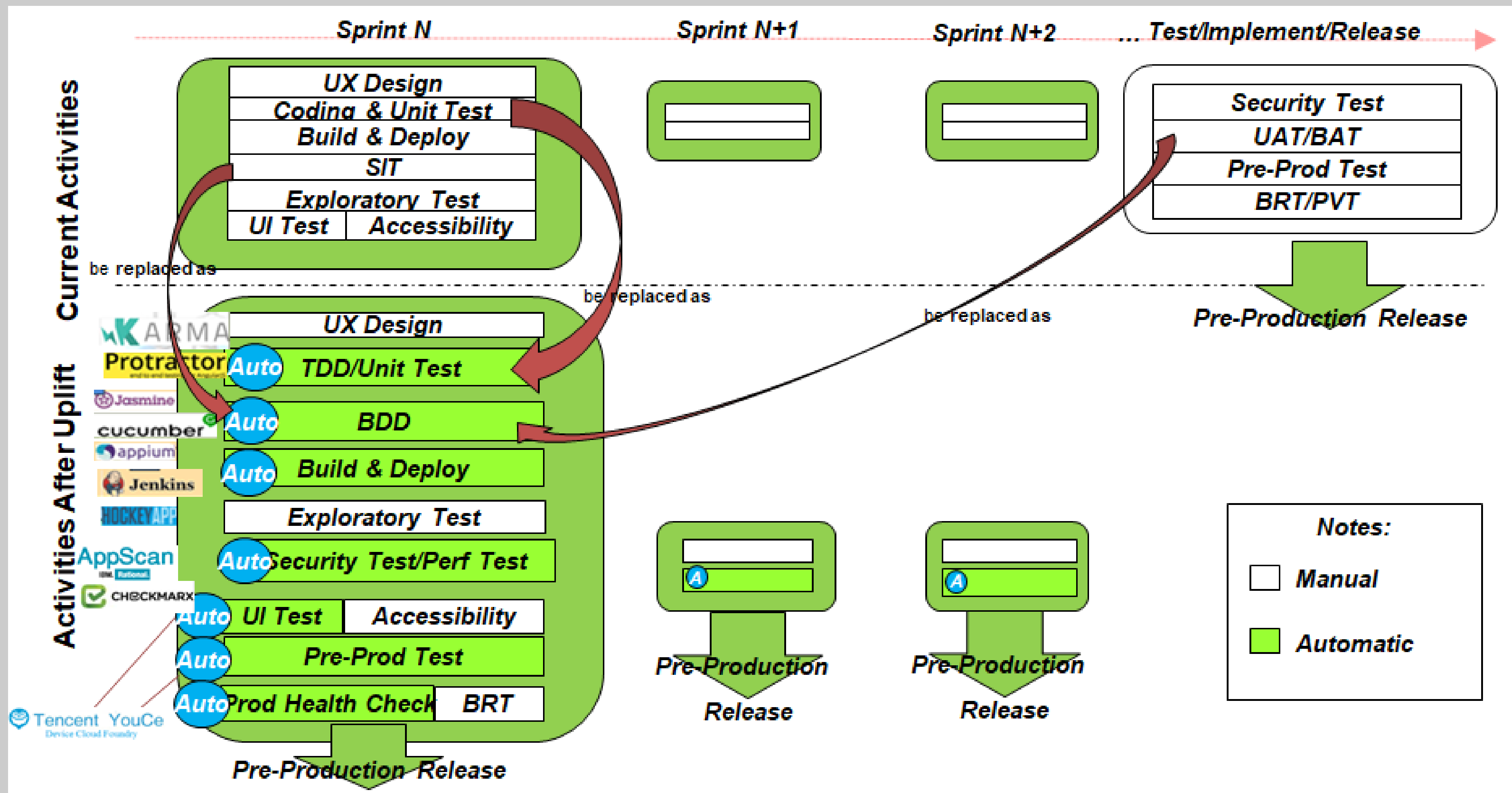
Imagine orchestrating global product development and deployment in over 16 key markets within just 6 months. It's a challenge I embraced head-on and conquered!

Join me as I reveal how we've harnessed the power of Scrum@Scale, Scaled Agile Framework (SAFe), and Behavior-Driven Development (BDD) to manage large-scale programs with over 10 internal and external teams.

Using **OpenAI** as platform to study and apply optimization via the whole ALM(Application Lifecycle Management):

- **6** roles joined together the study with Engineer, Tester, DevOps, Security, BA, Scrum Master and Solution Architecture
- Picked up 10+ DoD(e.g. Coding, testing) items for the study of AI implementation
- **10~40% saving per role** is realized.
- All people are using AI as tools for daily work.

Our Study from Scrum



The teams are able to shift-left.
What is the next step?

Using AI to accelerate correspondent DoDs

Current DoDs

Manual

e.g. Story writing

Manual/Auto

e.g. BDD

Auto

e.g. CI/CD

Future DoDs

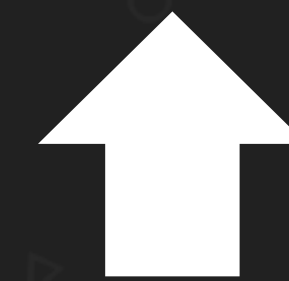
AI/Manual

AI/Manual/Auto

Auto

AI
(with prompt/SDL)

10~70% improvement



Previous	Day6	Day7	Day8	Day9	Day10
UI Design	Design Done				
BA	Story Prepare		Grooming and update story		
Technical Designer			Design (UML Diagram)		

No or low productivity improvement
 Significant productive improvement
 Still evaluating & PoC

- POC 1: AC Generation (from story / epic)
- POC 2: Plant UML Generation (from story / epic)

Current	Day1	Day2	Day3	Day4	Day5	Day6	Day7	Day8	Day9	Day10
Developer	Coding			Refactoring Code						
				Defect Fix			Unit Test			
SIT Tester	BDD Dev		Run on CI		Debug and Refactor case according SIT CI testing report					
UAT Tester						Review SIT CI testing Report				Sign off






- POC 3: Code Generation
- POC 4: Unit Test Generation
- POC 5: BDD Generation

Taking DoDs forward to seeing how AI helps

POC with OpenAI(1)

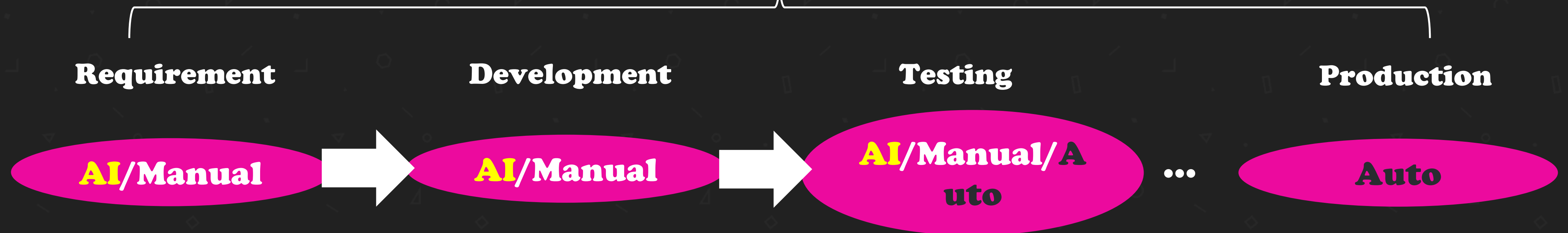
#	POC Tries	Cases	Output	OpenAI vs Manual
1	AC Generation (from story)	User Story to Acceptance Criteria	OpenAI is lack of bank knowledge and HSBC specified features / context.	Blank new story creation - Not helpful ❌ Supplementary ACs 10~20% ↑
2	Plant UML Generation (from story / epic)	User Story/ies to UML diagram	User story: most user stories will not describe the e2e application flow, so OpenAI don't know how to generate the sequence diagram; Most user story only focus on the one component's behavior (e.g. API user story), In fact, often the high sequence diagram came first and then the user story	Not helpful ❌
		Designer detail descriptions to UML diagram	Accuracy: 70%, Completion: 80% Most code which finally generated by OpenAI can be used directly; OpenAI can do most of the work if we input a very detailed description	around 1:1 (30mins vs 30mins) ❌
3	Code Generation	Story(API) to Java Code	Accuracy: 60%, Completion: 20% The verification mentioned in the user story can basically be done Most of the code generated by OpenAI can be used directly, but it needs to be carefully checked to avoid omissions. Classes and methods generated by OpenAI can improve efficiency in certain procedures The code generated based on User Story is limited, and more business codes need to be generated through other information input(e.g. API spec, database design), depends on story qualities	2-3mins + 10-15mins vs 1hours 70% ↑ Will further try more stories ✅
		Story(non-API) to Java Code	OpenAI lacks the context to understand the functions that need to be implemented, so OpenAI can only help generate pseudocode	Not helpful ❌
		Design detail descriptions to Java	Accuracy: >60%, Completion: >80% OpenAI can automatically recommend appropriate frameworks and generate reference-worthy code	20mins input + 20mins fine-tune vs 3 hours study & PoC 75% ↑ ✅

POC with OpenAI(2)

#	POC	Cases	Output	OpenAI vs Manual
3	Code Generation	UI Code generation with indicated UI image + OCR Technology	We applied embedding of UI standards of the bank to <u>ChatGPT</u> and using SDL to indicate UI element position. On top of this, OCR engine is used to cooperate with GPT to recognize UI image with key words. User just need to spend 10~15 minutes to prepare a simple image to GPT for generating UI page as code.	The code generation accuracy rate achieves > 90% and effort saving of writing code >40%.  
4	Unit Test Generation	Java & API (<u>Springboot</u> , Mule)	When GPT generates Unit Test with mock data preparation , GPT takes 20 minutes for testing and 15 minutes manual debug. If all go with manual approach, it takes 2 hours.	<ul style="list-style-type: none"> Simple (<1000 tokens): 10min vs 30mins, coverage >75% 65% ↑  Medium (1000 ~ 3000 tokens): 1hours vs 2hours, coverage >50% 50% ↑ Complex (>3000 tokens): >4hours vs > 4hours, coverage <50%, further try more prompts
		React	GPT may not be able to cover the Unit test for exception blocks. the test <u>coverate</u> is 71.43%. Other than this case, GPT could help to generate Unit test with almost 100% coverage >60% effort could be saved when using GPT comparing with manual approach. Manual: 10mins (write -> Try run -> Check Flow miss -> Write -> Try run (...) -> Done) GPT: 2mins (Generate + adjust) For complexity high (e.g. over 50K lines of code) coding, GPT may not help too much in terms of accuracy.	<ul style="list-style-type: none"> Simplest (<300 tokens): 3mins vs 10mins, coverage >90% 70% ↑  Simple (<1000 tokens): 10min vs 30mins, coverage >75% 65% ↑ Medium (1000 ~ 3000 tokens): > 4hours vs >4hours, coverage <50%, further try more prompts
5	<u>BDD Generation</u>	BDD Feature files BDD auto java scripts	Using <u>OpenAI</u> can save time for BDD testing for sure. It provide high accuracy (around 80%) still can improve about 50% efficiency. However, the code it generates still needs to be amended and debug to fit in our own testing framework, and the users need to be familiar with the questioning method. Few-shot examples can be very useful in some cases. <u>High quality user story (AC) is required to improve efficiency.</u>	Simplest case (login): 15mins vs 30mins 50% ↑  Simple cases (input page) 30mins vs 1 hour Will further try more for complex cases.

Next step

Using **Chatbot** to connect all uplifted DoDs to optimize full Application Lifecycle Management(ALM) -> Transform as Low Code NO Code



AI, I need... : **Human**

AI: You have the code here:

<Div id=abc....>

AI, please share with me the test report: **Human**

AI: You have the test result:

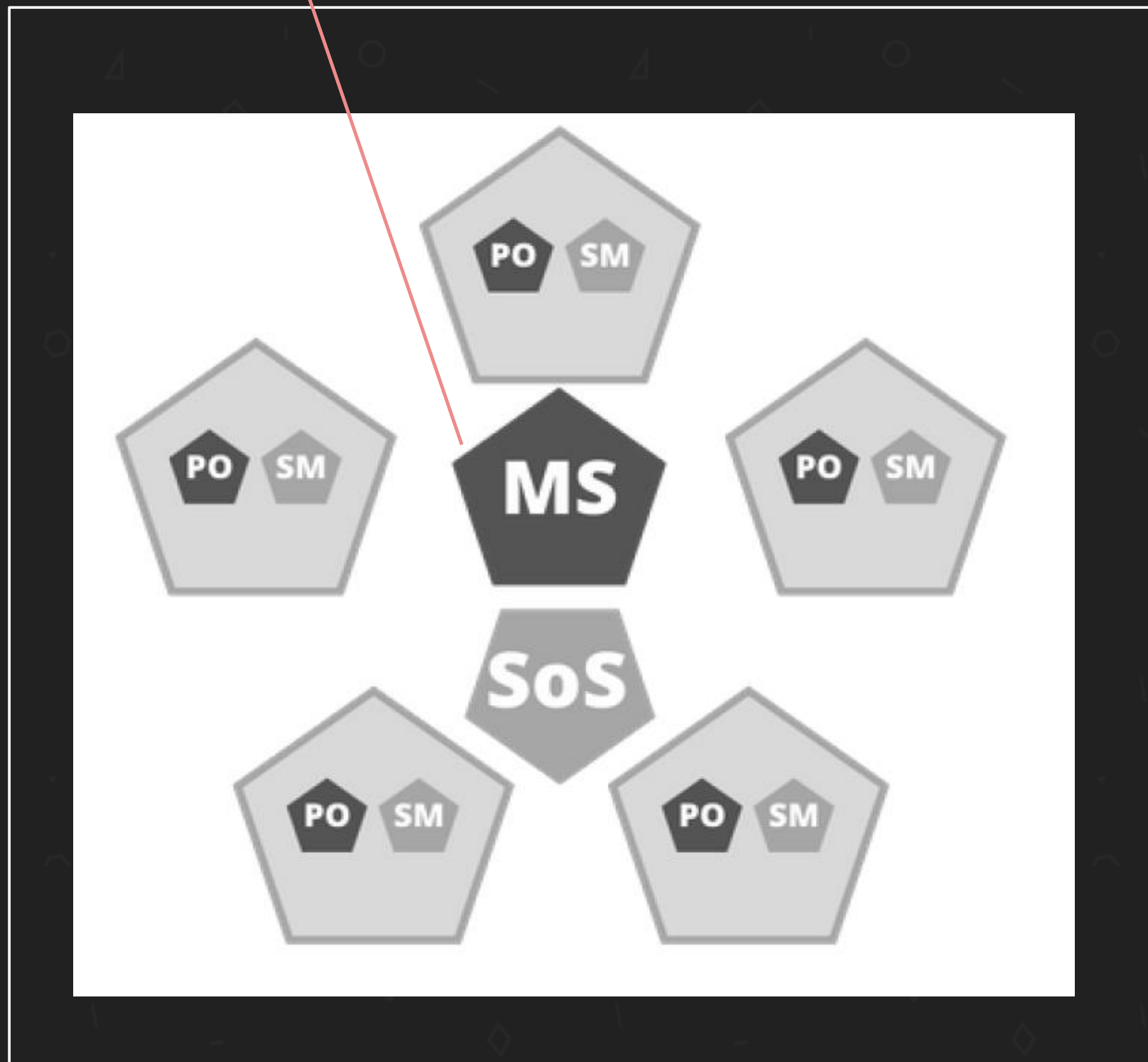
Unit test coverage: 80%, BDD passing rate: 90% ...

Please help to release above source code to production.: **Human**

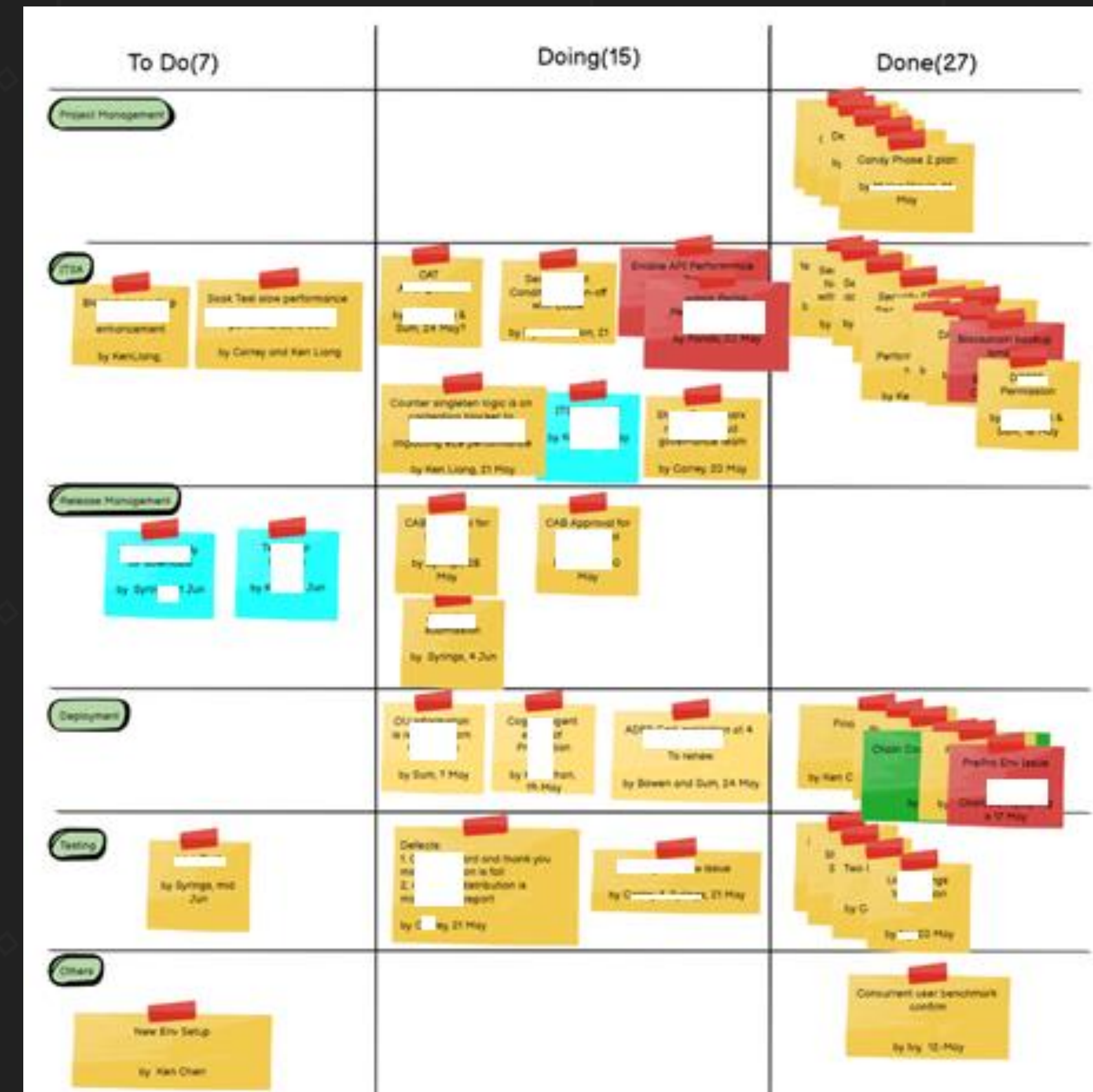
AI: Done on B site with 10% loading, monitoring with canary release model

Scrum of Scrums and Meta Scrum Team Implementation

Scrum@Scale will continuously help for the project delivery after AI driven development uplift of team level. We will just be with less of workers.



Scaled Daily Scrum of SoS or MS Kanban



THANK
YOU