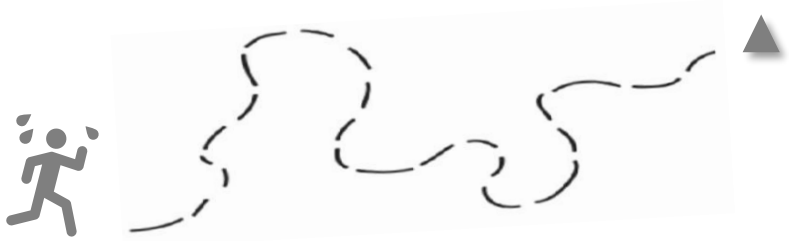
人工智能與敏捷開發

- Ruddy Lee

人工智能時代的工程師難為

理論上；工程師應該要能夠理解他們正在開發的軟體中所有實作細節，但這將只是一個崇高的目標。因為在雲原生時代，開發人員面臨著越來越多的工具、技術、思維方式的選擇，這給我們帶來了極大的認知負荷和工作量。



而對真實世界的開發工作而言，
真正的關鍵應該是如何讓工程師 在需要時能夠輕鬆了解實作的細節，進而針對核心作業去完成開發工作，這是 開發者體驗 (DevX) 的目標。

工程師

我們需要

- 一個發掘與回答問題的地方
- 一個讓工程師看見成果與記錄功能使用率的地方
- 一個省去需要繁瑣步驟，才能夠快速取得環境設置的功能
- 一個讓我們能夠盡快佈署試錯的地方
- …
-

環境

工程師

我們需要

- 一個發掘與回答問題的地方
- 一個讓工程師看見成果與記錄功能使用率的地方
- 一個省去需要繁瑣步驟，才能夠快速取得環境設置的功能
- 一個讓我們能夠盡快佈署試錯的地方
- **一個知道原由，尋找疑問、解答的地方**
- **一個(團隊)發現問題，回復，解釋、解答的地方**
- **...**
-

環境

原因

工程師

我們需要

- 一個發掘與回答問題的地方
- 一個讓工程師看見成果與記錄功能使用率的地方
- 一個省去需要繁瑣步驟，能夠快速取得環境設置的功能
- 一個讓我們能夠快速佈署試錯的地方
- 一個尋找疑問、解答的地方
- 一個(團隊)發現問題，回復，解釋、解答的地方
- 一個能夠看到彼此狀態並分享知識的地方

…

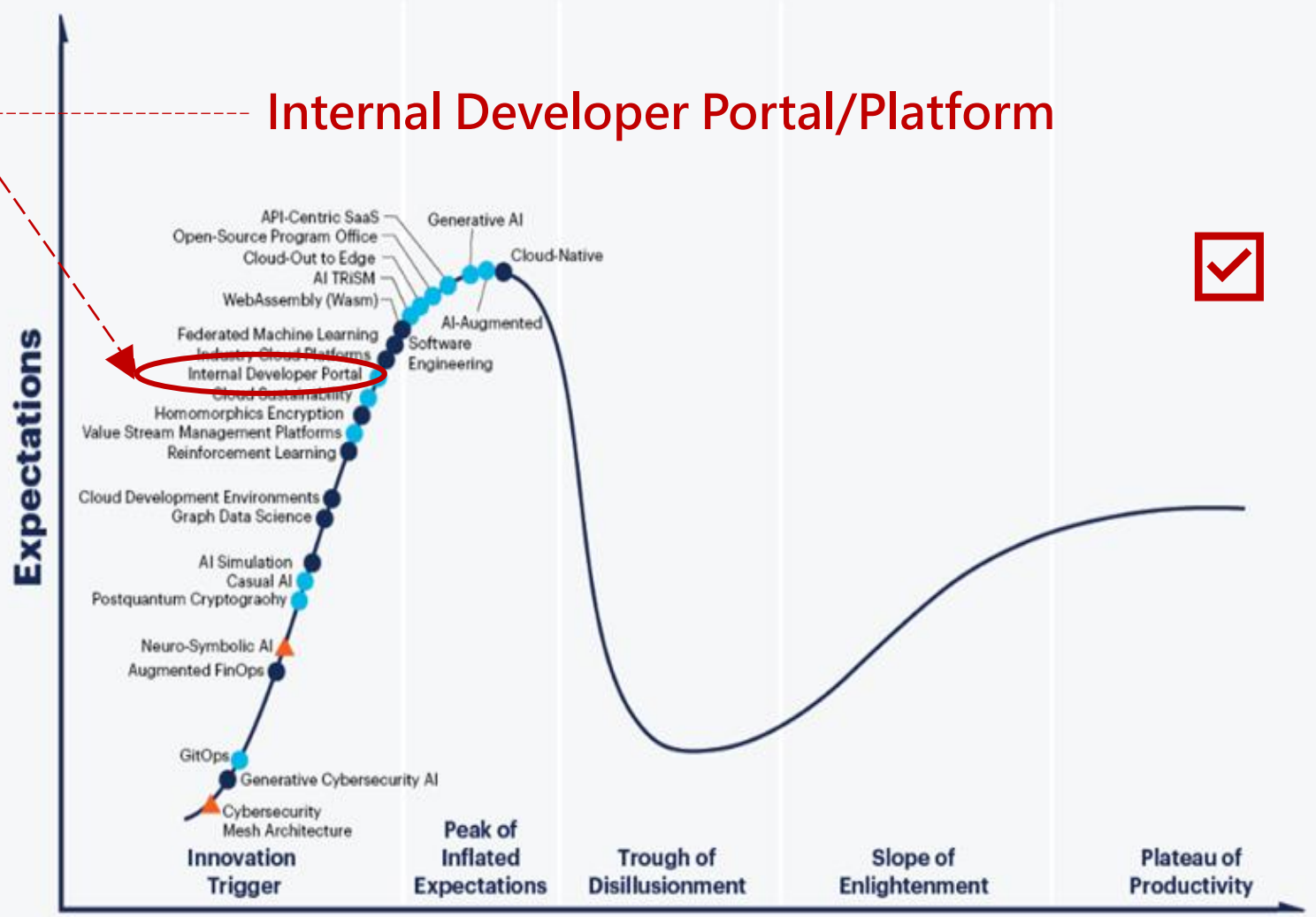
- 一個資深工程師能夠展現深度(隱性)知識的地方

環境
原因
方式
內容
人員

Gartner 發佈
新興技術曲線圖

Hype Cycle for Emerging Technologies, 2023

Internal Developer Portal/Platform



2 to 5 years
5 to 10 years
> 10 years

階段

技術萌芽 過度擁簇 幾乎滑落 均衡成熟 生產力平台

四個主題

- 新興人工智慧、
- 開發人員體驗 DevX、
- 原生雲端、
- 以人為中心的安全和隱私

gartner.com

Gartner

內部開發者平臺

Internal Developer Platform (IDP)

IDP 是一套由平臺工程團隊維護的工具和技術，讓開發者能夠更快更便捷地構建和部署軟體。

An Internal Developer Platform (IDP) is built by a platform team to build golden paths and enable developer self-service.

黃金路徑 (Golden Paths) : 是一種預先設置的架構和支持方法，用於在 IDP上構建和部署特定類型的軟體，是指導開發人員如何使用 IDP 的最佳實踐。

IDP的目的

通過抽象化基礎設施的細節和提供一致、標準化的工作方式來減少軟體開發的複雜性和運營成本。

服務開發者

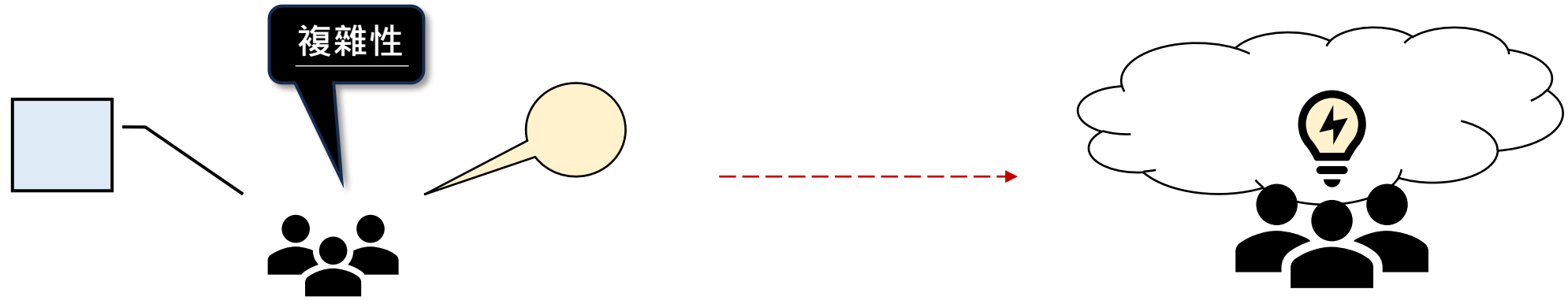
劃分複雜性

有效的內部開發人員平台

Thinning Viable Portal /Platform

從本質上講，**有效的內部開發人員平台**可以**劃分複雜性**。
團隊或組織內，每個人都有自己擅長處理的**複雜小領域**，而其他人可以安全地忽略這些領域。

每個人都根據一般 (general)的抽象化來處理自己的實作細節，並了解如何將所有內容如何組合在一起。



知識體系

內部開發者入口網站
Internal developer portals

是一個用於組織內部開發者團隊的重要工具，目的在幫助開發者能夠更有效地 **查詢**、**使用** 和 **分享** 內部的開發資源、工具 和 文件資料。

- 易用性 Usability
- 文件和教育資源
- 訪問權限和身份驗證
- 開發工具
- API 管理和測試工具
- 發布版本控制系統
- 通訊和社交工具
- 性能監控和分析
- 安全性 Security
- 度量和反饋

Cognitive load

減少工程師的認知負荷

認知負荷
組織可以做甚麼?

瞭解工程師的負荷

負荷: 工程師工作記憶中使用的腦力勞動

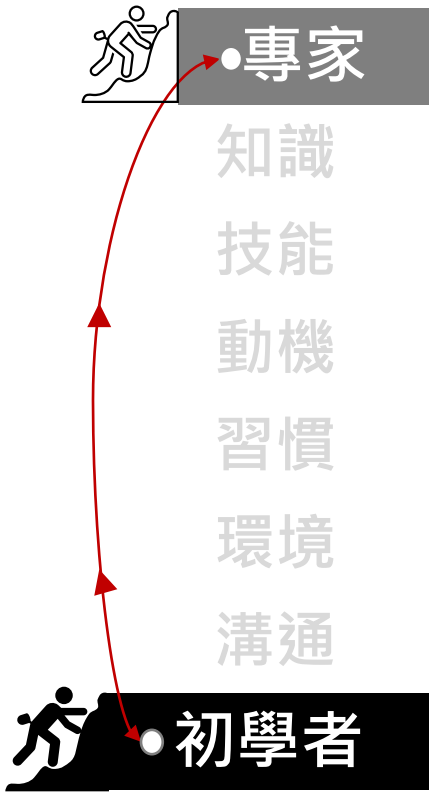
cognitive load
- John Sweller

1. **固有 認知負荷**
   - 與問題領域的基本任務相關。

   組織應該試圖最小化固有認知負荷，例如: 上任、培訓 ...來快速實現。
2. **額外 認知負荷**
   - 與處裡的環境相關。

   應該儘量消除**額外認知負荷**。
3. **相關 認知負荷**
   - 與那些需要格外關注學習和高性能方面的任務相關。

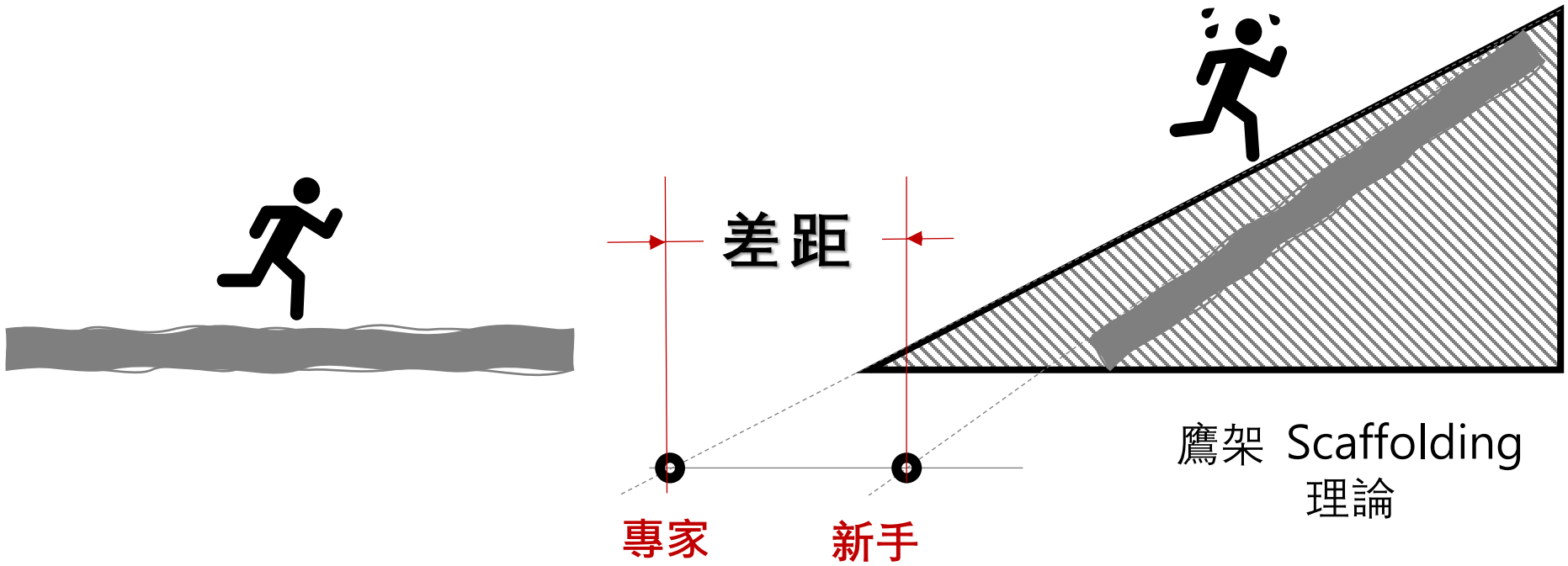
   為相關認知負荷預留更多空間，才是一種讓團隊增值的思維。



「**複雜度** 是認知負荷的一大因素」，讓團隊聚焦在單一領域這是 **微服務化** 的效用。

你可以怎麼做?

搭建**鷹架**來協助開發者克服障礙

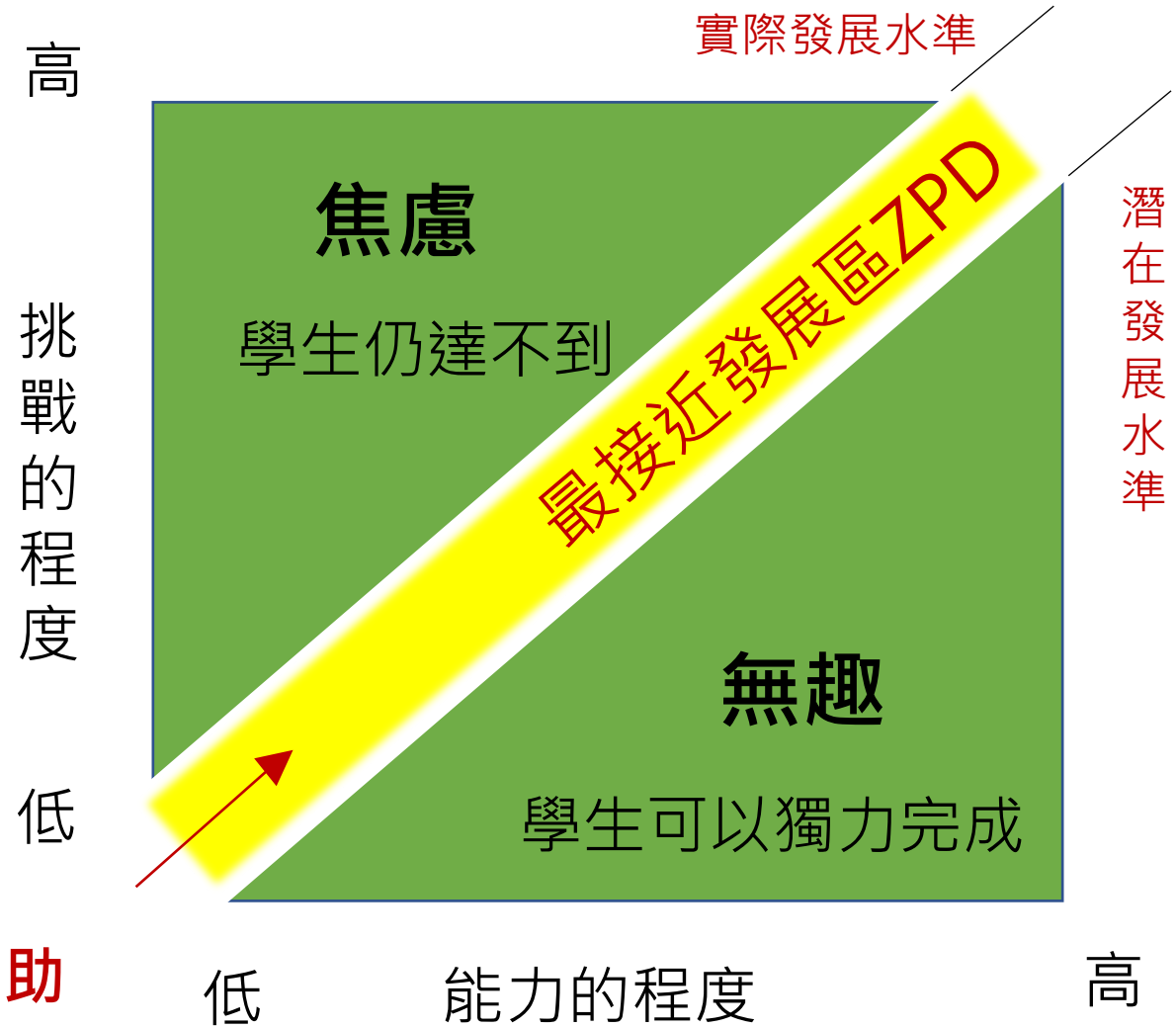


鷹架 Scaffolding
理論

情境學習理論

鷹架理論

By: 維高斯基 Vyzotsky

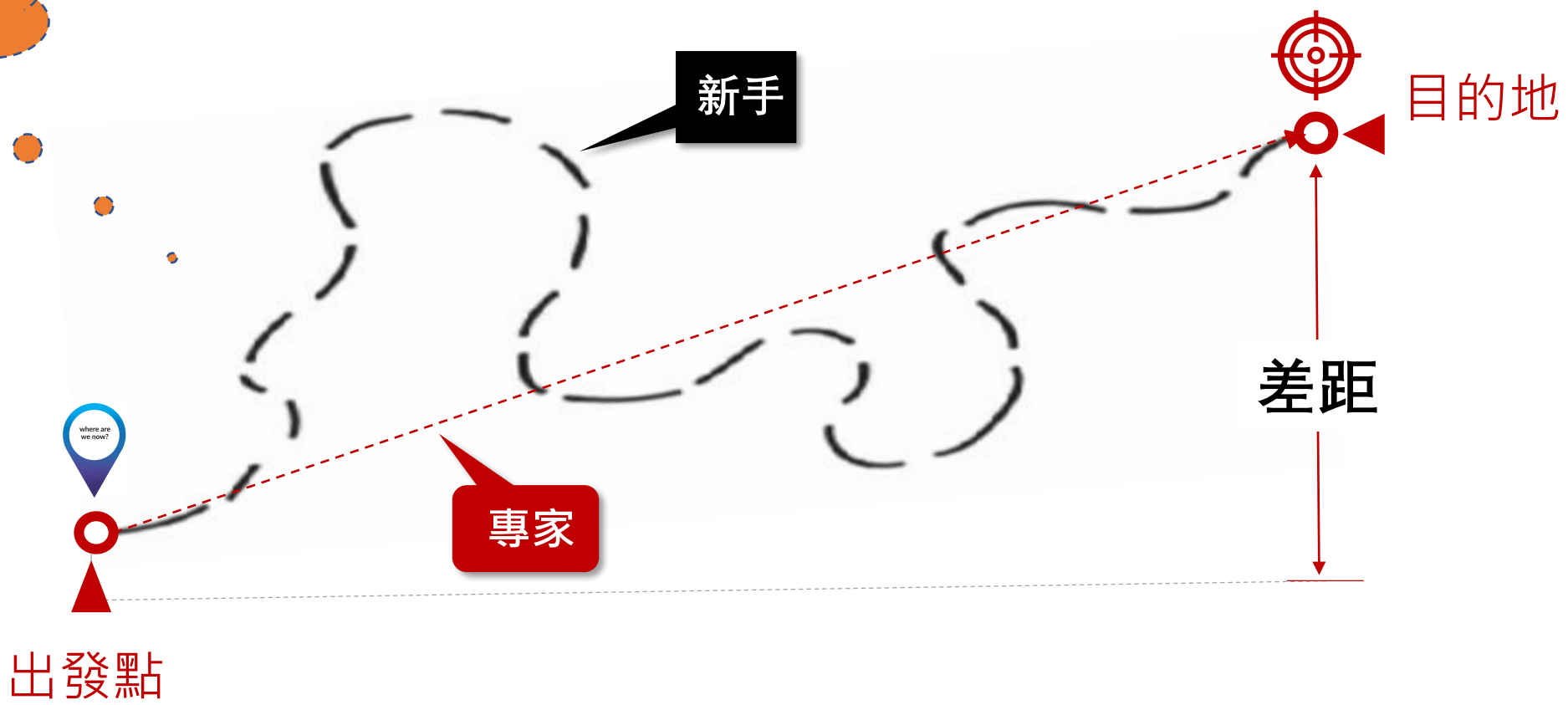


新手可以在幫助下達到的區域

要學會什麼?

開發者體驗設計的過程

形成概念

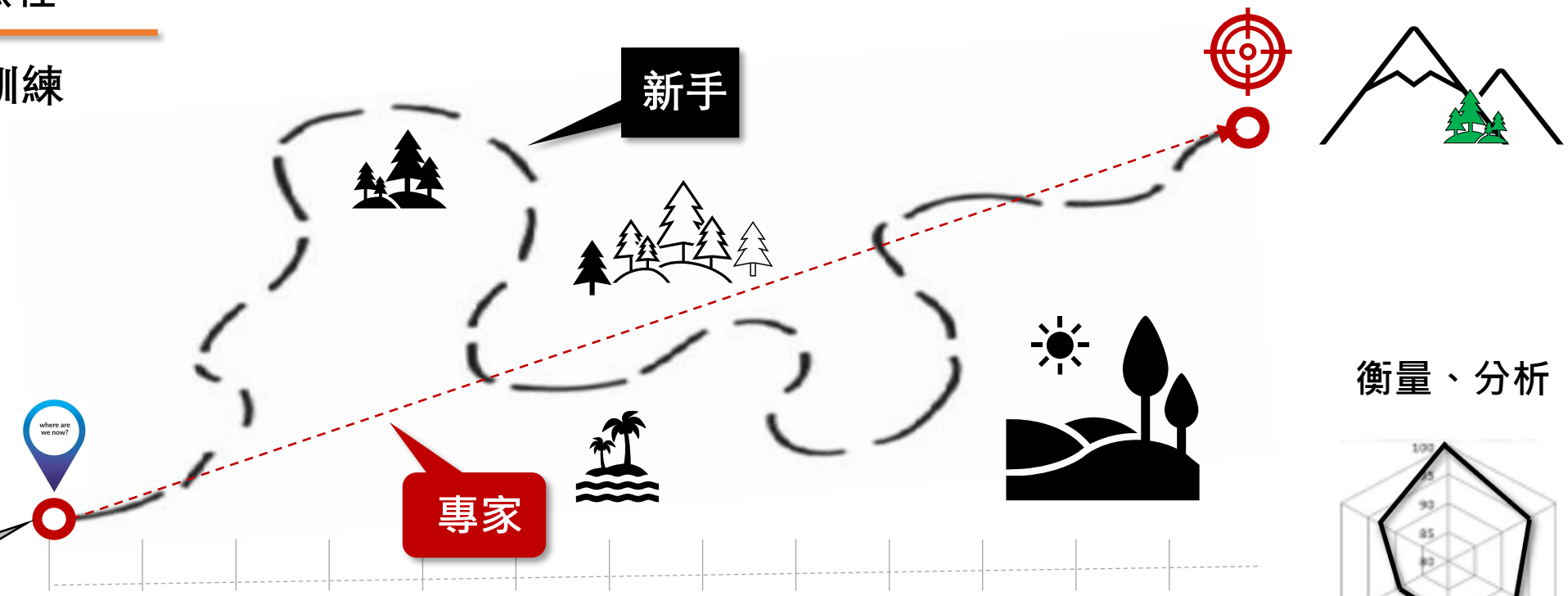


要學會什麼?

將過程視為一段旅程

如果開發是一段旅程

- 必須配備、行前訓練
- 還需要什麼?
- 有了這些東西，就能快速到達嗎?



衡量開發者當前的技能水準

衡量、分析

雷達圖分析

體能訓練、搜集資訊、 ...

如何彌補差距?

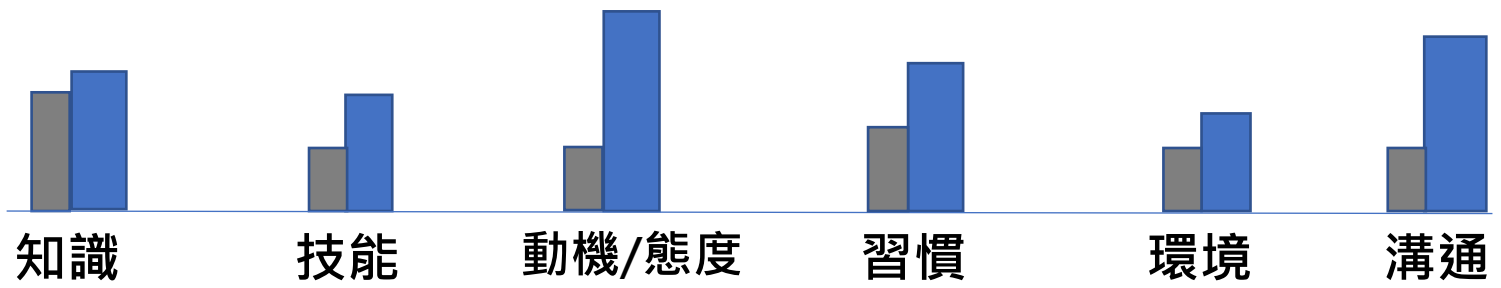
思考如何彌補差距

新手

專家

回顧與開發
經驗的轉化

團隊距離達成
任務的差距



認知差距

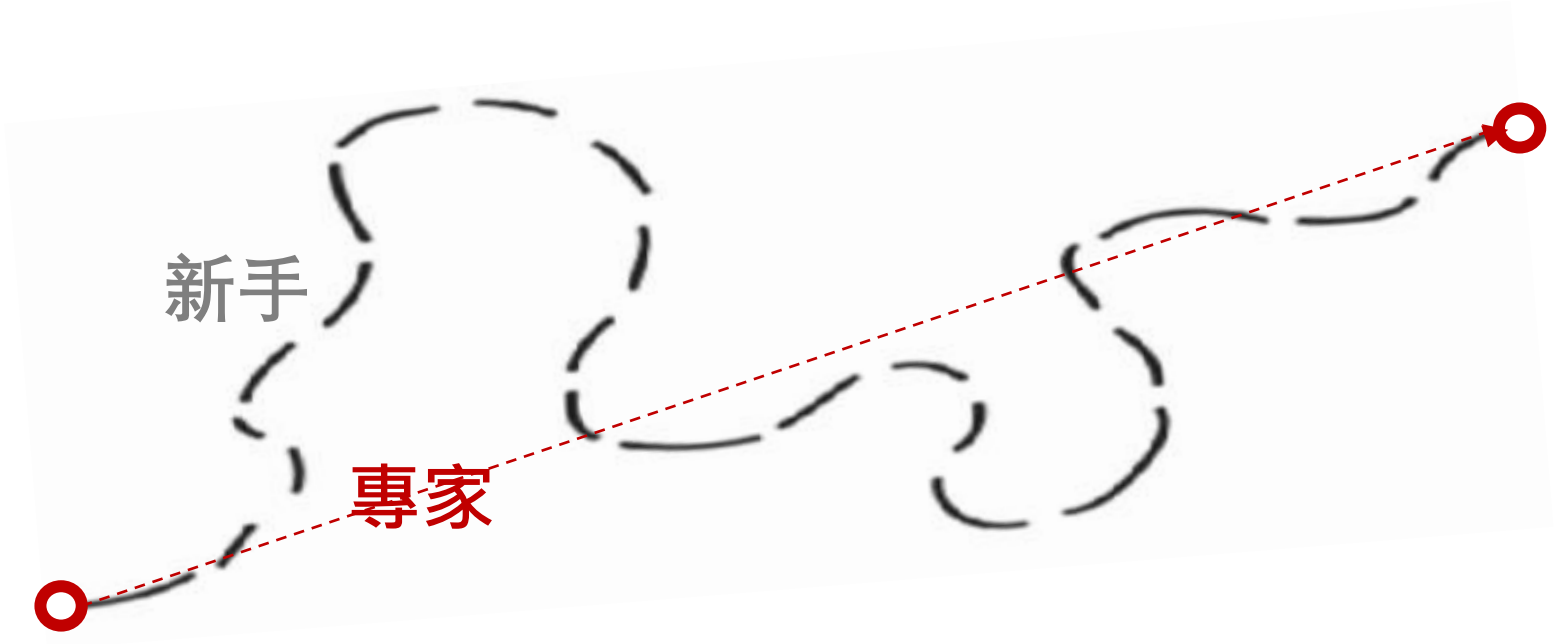
朱莉·德克森

幫助學習

IDP
內部開發者平台

解決方法

建置平臺來協助開發者



AI
鷹架

調整
持續衡量

TVP, Thinnest Viable Platform
- 馬修·斯凱爾頓

認知差距

最小可行平臺，IDP

知識　技能　動機/態度　溝通　環境　習慣

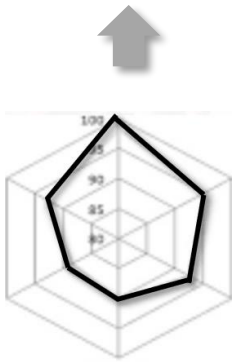
協作、服務、引導

DevX 即 平台

團隊的 Copilot

團隊的 ChatGPT

IDP, 內部開發人員平台
Internal Developer Platform



調整

持續衡量

協作、服務、引導

AI 最小可行平臺 Self service

知識 技能 動機/態度 溝通 環境 習慣

TVP, Thinnest Viable Platform

- 馬修·斯凱爾頓

認知差距

開發時期的
文件、資源

開發時期的雲端
建置、調整、維運

以設計平臺工程的方式，減輕開發團隊的「**額外認知負荷**」。

Internal Developer Platform

內部開發者平台 IDP

- 專屬於平台、Ops及DevOps團隊，並由平台團隊所構建，
- 用於建立及提供各種資源的黃金路徑並實現開發者自助服務 self service。
- IDP 由許多不同的技術和工具組成，它們以降低開發人員認知負荷的方式粘合在一起，而無需抽像的上下文或 底層技術。
- 平台團隊遵循最佳實踐，將其平台視為產品，並根據使用者研究進行建置、維護和持續改進。

內部開發者平台 IDP， 是一個用於協助內部開發團隊更高效地建立、測試和部署軟體應用程式的工具或系統。它的功能應該能夠支持開發生命週期的各個階段，並提供開發團隊所需的工具和資源。目的是降低開發者的**額外負荷**。

認知負荷

瞭解工程師的負荷

系統的「複雜度」是認知負荷的一大因素，

讓團隊聚焦在單一領域這是 的效用(**微服務化**)。

小增量、多迭代、尋求回饋

認知負荷

開發速度 與 開發團隊 的認知負荷

你的團隊目前的開發速度如何?

他們感覺很興奮還是很疲倦?

. . .

覺得學到很多東西?

還是沒有學到東西?

Sprint 的負荷

團隊開發速度的考慮

短期速度

長期速度

> 12hr

快節奏
衝刺

6-8hr

Marathon
慢跑式

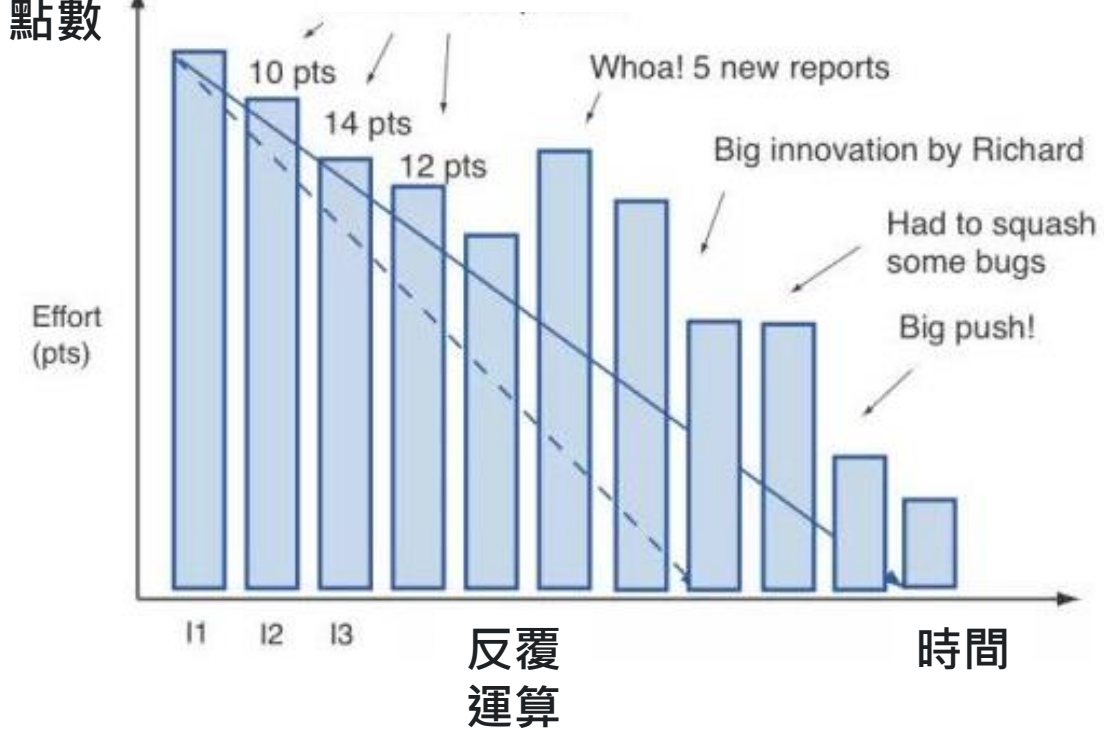
方式

目標

學習

狀態

我們的開發速度不如預期



團隊開發速度的考慮

開發者體驗

	短期速度	長期速度
方式	> 12hr **快節奏 衝刺**	6-8hr **Marathon 慢跑式**
目標	開發人員的 可容忍程度 **DevOps追求極速**	Marathon的意義 以盡可能快的速 度跑很長的距離
學習	**大量學習**	**持續學習**
狀態	**很快倦怠**	**產能不足但 穩定成長**

團隊開發速度的考慮

開發者體驗

	短期速度	8-10hr	長期速度
方式	> 12hr **快節奏衝刺**	**中等衝刺**	6-8hr **Marathon慢跑式**
目標	開發人員的可容忍程度 **DevOps追求極速**	持續多年成為習慣	Marathon的意義 以盡可能快的速度跑很長的距離
學習	**大量學習**	**事業家庭兼顧**	**持續學習**
狀態	很快倦怠	**少動力 學習趨緩 生產力下降**	產能不足但穩定成長

團隊開發速度的考慮

短期速度 ——— 長期速度

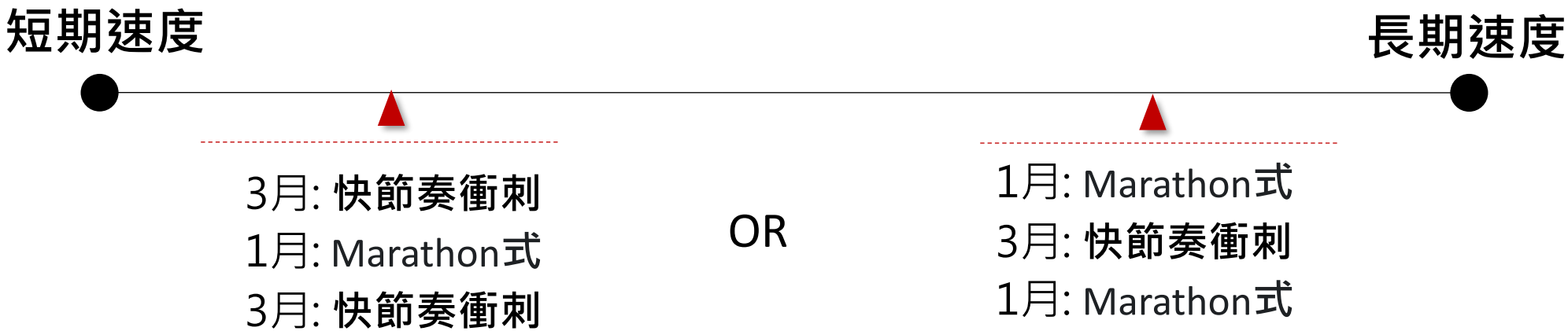
	> 12hr	Intervals	8-10hr	6-8hr
方式	**快節奏衝刺**	**區間式**	**中等衝刺**	**Marathon 慢跑式**
目標	開發人員的可容忍程度	3月: 快節奏衝刺 1月: Marathon式 3月: 快節奏衝刺	持續多年成為習慣	Marathon的意義 以盡可能快的速度跑很長的距離
學習	**大量學習**	**里程碑式學習**	**事業家庭兼顧**	**持續學習**
狀態	很快倦怠	**平均速度高於中等衝刺模式**	少動力 學習趨緩 生產力下降	產能不足但穩定成長

團隊開發速度的考慮

短期速度 ⟷ 長期速度

	> 12hr		8-10hr		6-8hr
方式	快節奏衝刺	Intervals 區間式	中等衝刺	Intervals 區間式	Marathon 慢跑式
目標	開發人員的可容忍程度 **DevOps追求極速**	3月: 快節奏衝刺 1月: Marathon式 3月: 快節奏衝刺	持續多年成為習慣	**1月: 快節奏衝刺** **3月: 馬拉松式** **1月: 快節奏衝刺**	Marathon的意義 以盡可能快的速度跑很長的距離
學習	**大量學習**	**里程碑式學習**	**事業家庭兼顧**	★ **目標式學習**	**持續學習**
狀態	很快倦怠	平均速度高於中等衝刺模式	少動力 學習趨緩 生產力下降	**平均速度高於Marathon模式**	產能不足但穩定成長

如何做到 Intervals 區間式 的開發模式

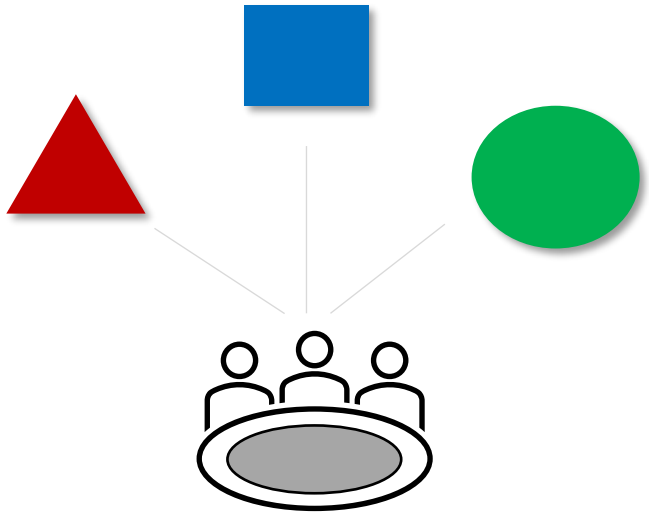


擬定軟體開發策略

- 依據上次犯過的錯誤擬定策略
- 痛點先行 Pain point first
- 開發流程更自動化
- 選擇業界的最佳實踐
- 避免重新發明輪子


謝謝!
下回見

遠端工作的挑戰



《 思維可視化 》

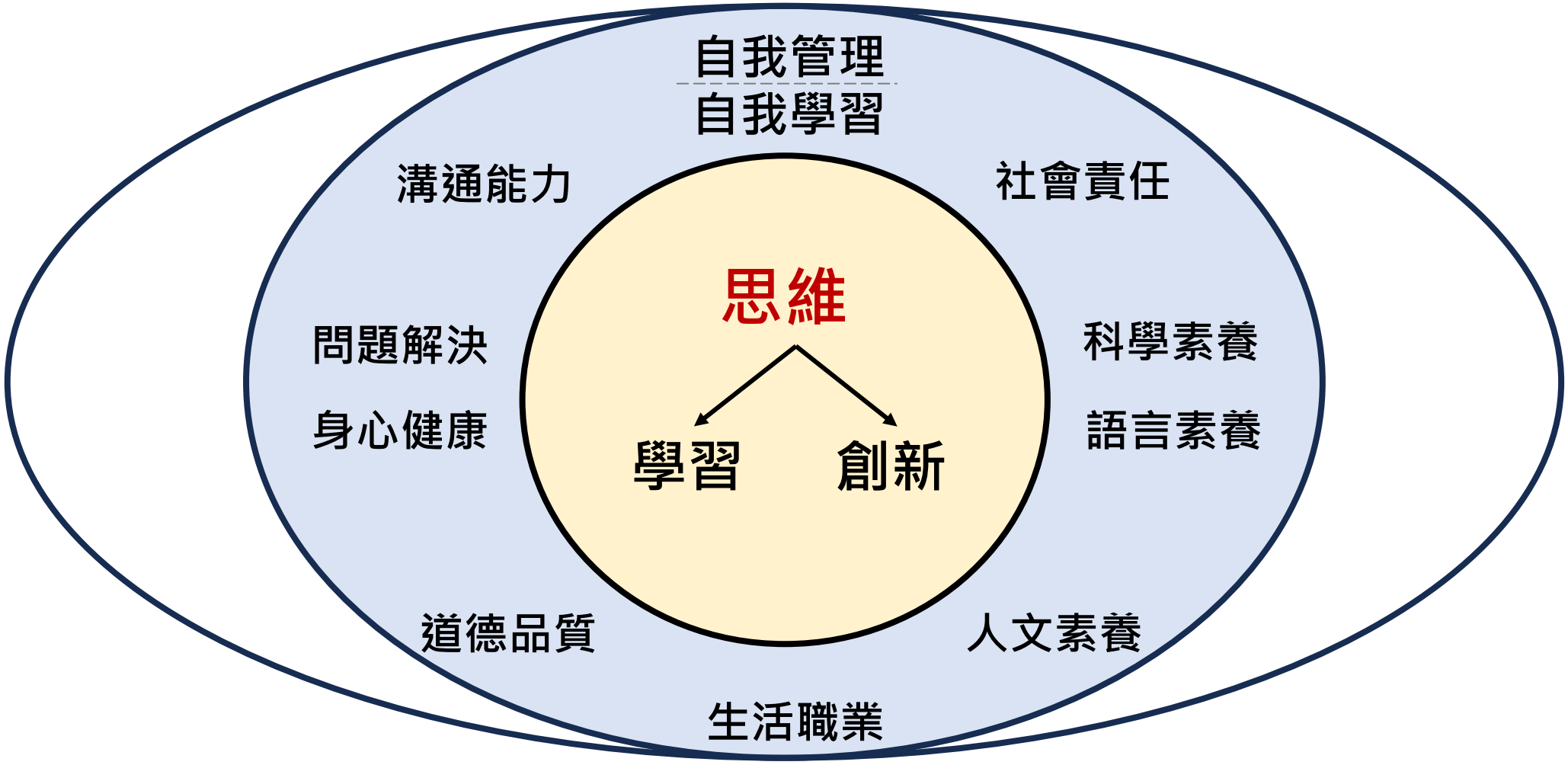
Thinking
Visible

ScrumMaste

解套遠端站立會議

DevX

工程師是怎麼想的 ...



如果我們相信**學習是思考的結果**，那麼我們不僅要讓我們的工程師去思考，而且要理解正在展開的思考過程，以便我們能夠支持它、促進它並發展它。

可視化思維 MTV

Making Thinking Visible

- **啟發性提問**

「**你為什麼這麼說？**」透過使用促啟發問題時，目標是瞭解學員的想法，進入他們的頭腦，使他們的想法顯現出來。

- **傾聽**

通過我們的傾聽，我們為學員提供了一個開放的空間，讓我們看到他們的思維與想法。

- **文件紀錄**

思考和學習的過程可能是難以捉摸和短暫的。文件記錄就是盡可能豐富地捕獲這個過程的工作。將隱性知識轉換為顯性知識的過程中記錄思維。

- **思維流程**

思維流程是讓思維視覺化的核心練習。它們是促進思維的工具，是揭示和支撐思維的結構，隨著時間的推移，它們的使用會逐漸成為一種固定的行為模式。


謝謝!
下回見

問題 與 回答

【問題】	【分析】	【回答】	【確認】	【完成】
10	3	1	3	
High ↑ ↓ Low	問題描述	解答	點頭	最後的回顧檢視
	加以重點描述	在二分鐘內做解答	提問者確認解答	

專案開始之初，首重看見全貌

一旦；當你把眼光投注在哪一個要項的時候，
實際上你就只看到那一部分，

你的思緒將被那一部分的內容所牽動，很難再看見其他的事...
所以我們要退後一步，

不! 有時要退後很多步，才能比較清晰地看見全貌。

- 請思考你的「**退後一步**」是什麼?

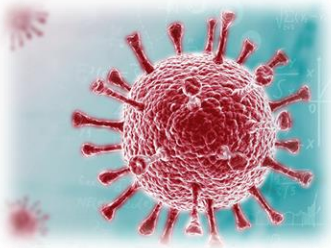
探索真正問題？

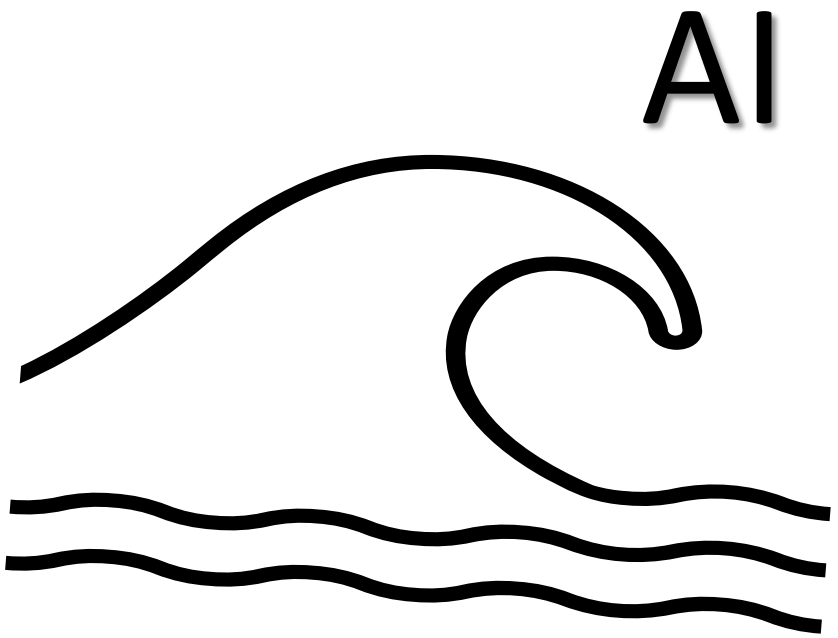
提問 ...

我在哪里?

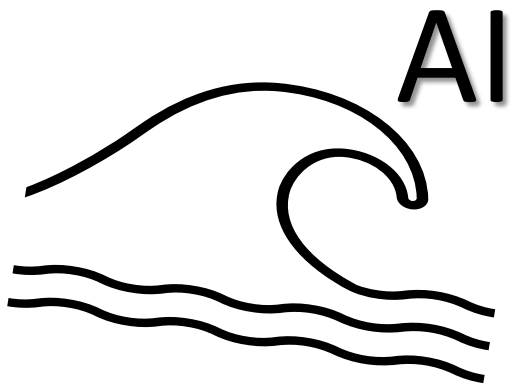
WE ARE HERE!

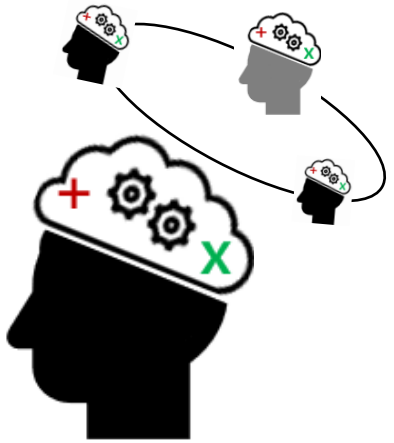
人工智慧正在滲透到當今所有行業的幾乎每個流程中





思維可視化是解決人工智慧衍生出來的教育問題

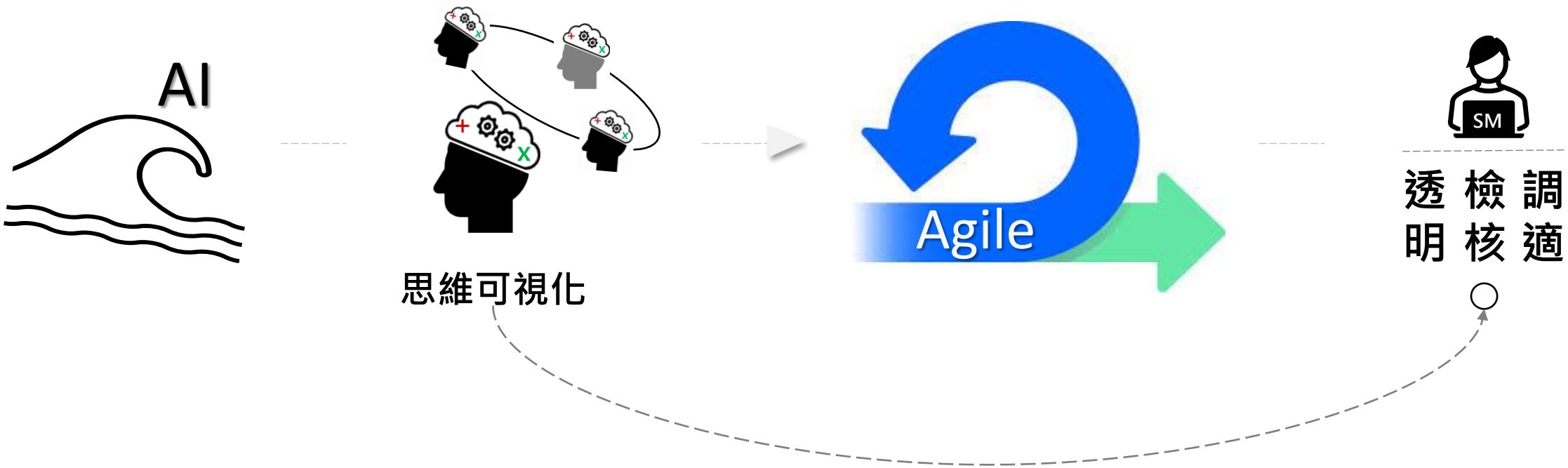




思維可視化

Make Thinking Visible

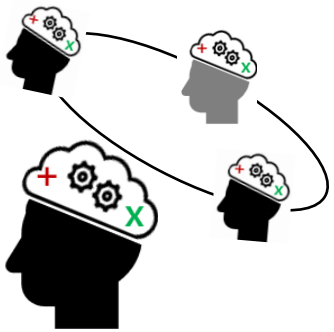
敏捷也受到 AI的衝擊



敘畫

將可視化思維運用在會議流程中

AI



思維可視化

Agile

思維
可視化

SM

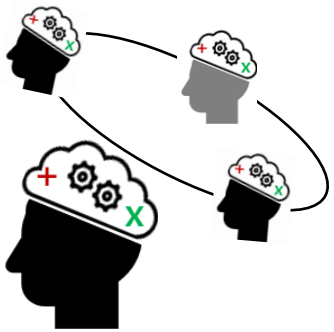
透 檢 調
明 核 適

團隊自組織

敘畫

序畫室可視化思維的流程

AI



思維可視化

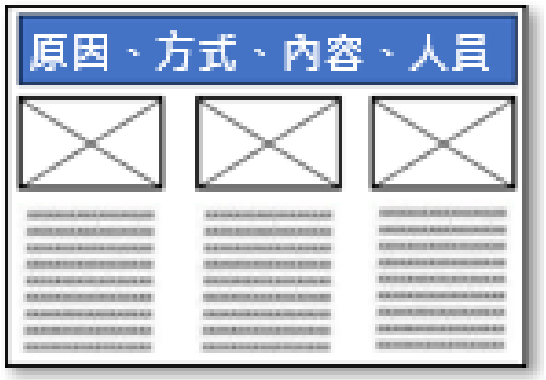
Agile

思維
可視化

SM

透 檢 調
明 核 適

Internal Developer Portal



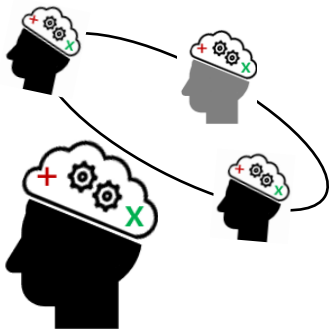
::

團隊自組織

敘畫

從人工智慧到開發者體驗

AI



思維可視化

Agile

思維
可視化

SM

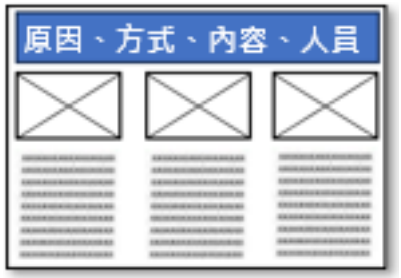
透明 檢核 調適

DevX

開發者體驗

::

Internal Developer Portal



::

團隊自組織

內部開發者平台 與 DevOps

可以幫助團隊實現敏捷開發中的原則和最佳化實踐

自動化和持續交付

內部開發者平台透過提供自動化工具和流程，幫助開發團隊實現持續交付。

開發環境的標準化

標準化了開發環境，使得所有團隊都能夠使用相似的工具和流程進行開發作業。

培訓和知識分享

提供了數據分析和監控工具，以幫助團隊了解應用程序的性能，並做出有關未來改進的決策。(開發功能的使用率)

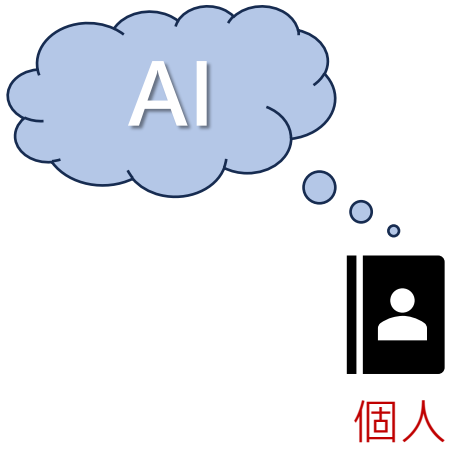
數據和監控

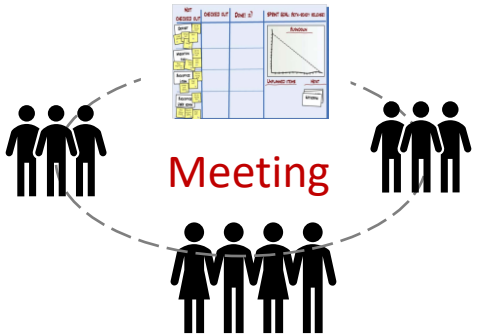
內部開發者平台包括文件、教育資源和知識庫，有助於敏捷團隊的培訓和知識分享。打破 scrum 所造成的知識 silo。

團隊自組織

自組織團隊如何運用 AI?

團隊共用AI還是個人個別運用AI？

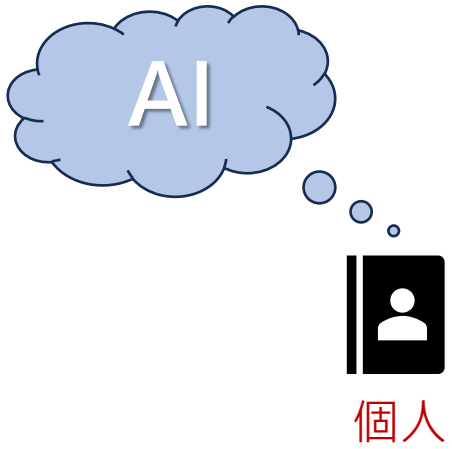




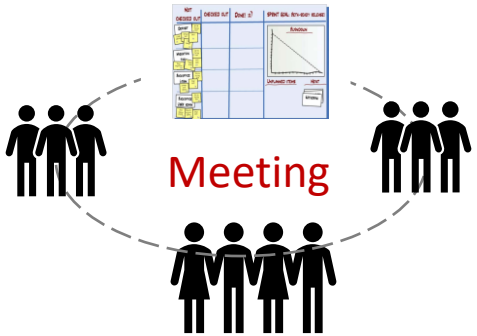
AI 對站立會議的影響？

AI 對敏捷會議的影響？

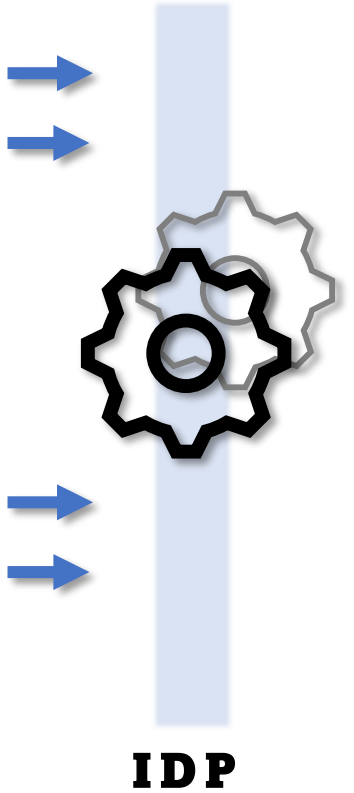
團隊如何運用 AI?



透明化



Internal Developer Portal/ Platform

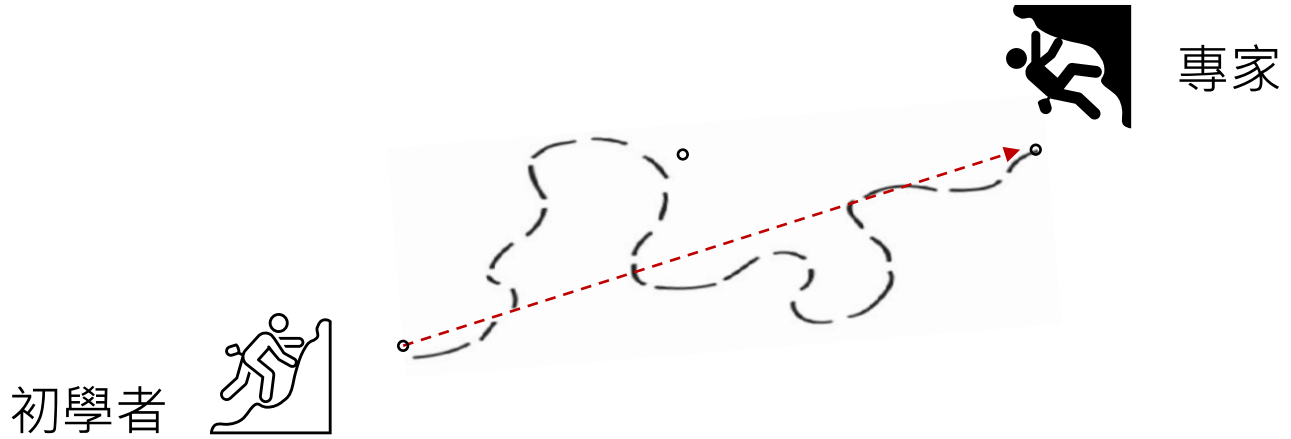


開發者體驗

IDP

AI 不是資深工程師

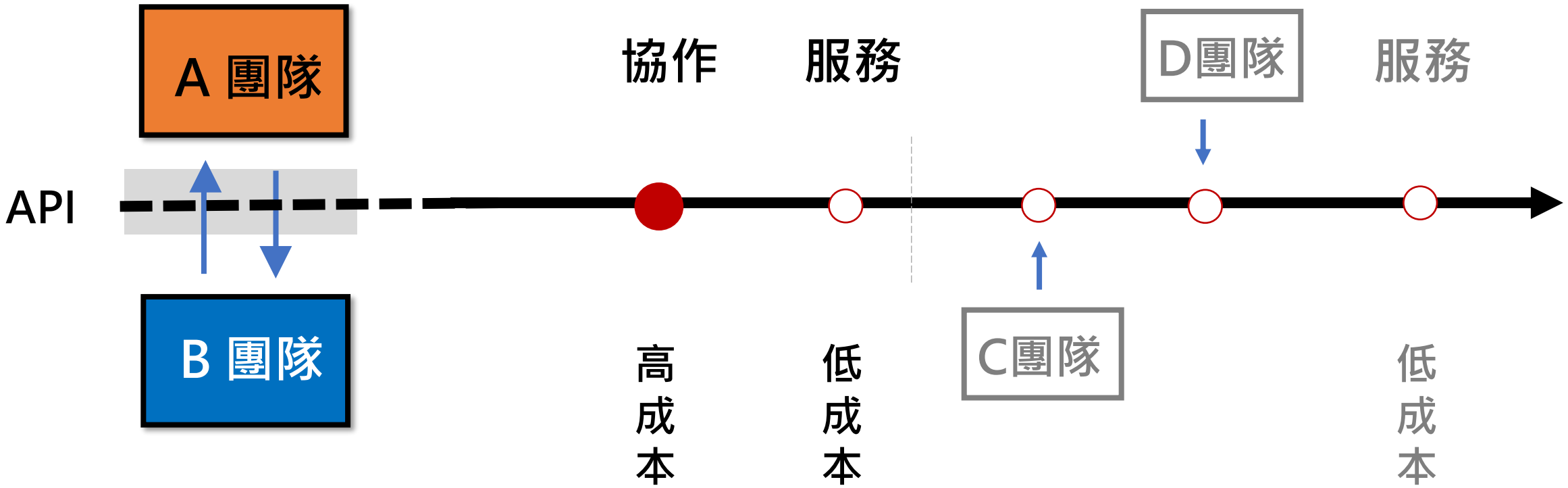
如何將資深工程師的經驗
由隱性知識轉換成顯性知識?



團隊互動方式的演化

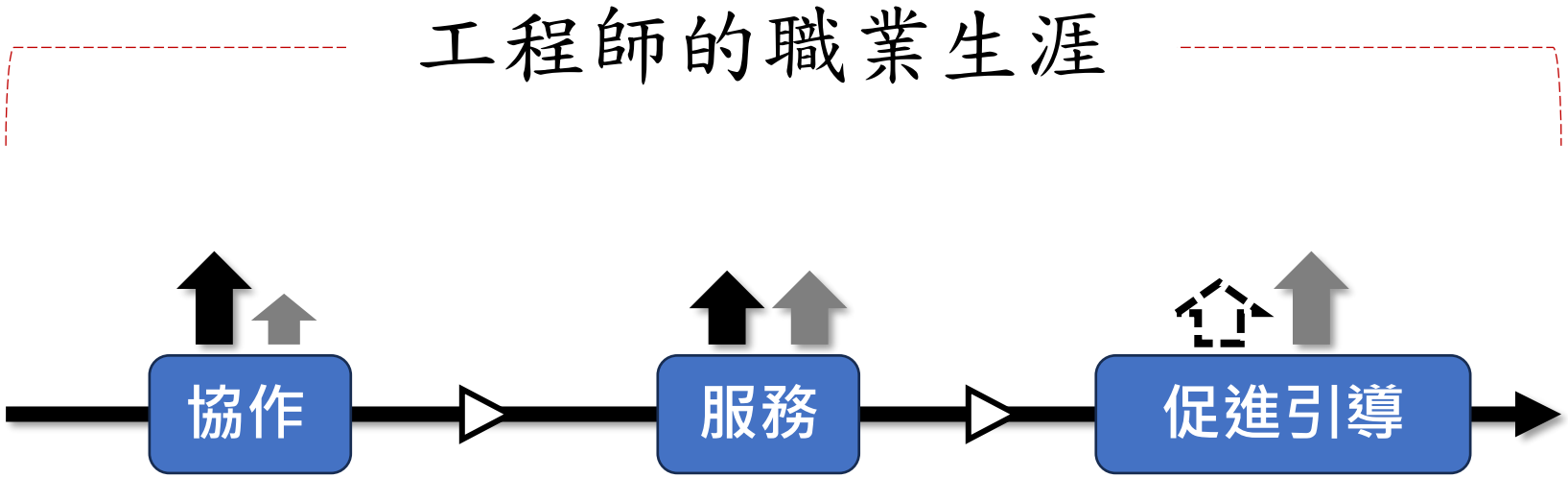
○ 思維視覺化

- 團隊交互由 協作 轉為 服務
- 轉化 溝通 為 促進引導



團隊間的互動行為

- 協作
- 服務
- 促進引導



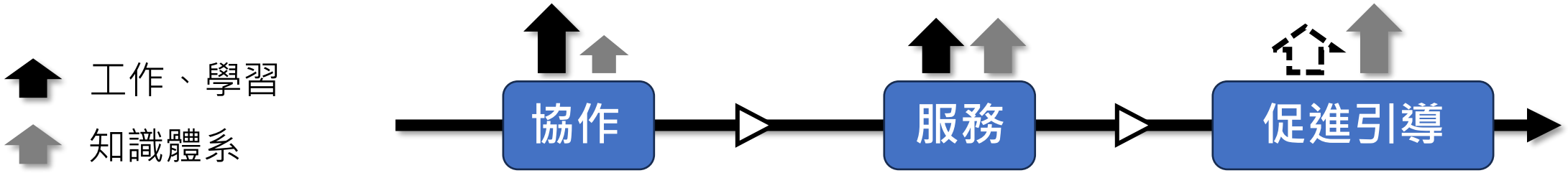
工作、學習

知識體系

團隊間的互動行為

- 協作
- 服務
- 促進引導

工程師的職業生涯



工程師要將開發專案時所習得的隱性知識轉化出來成為顯性知識，
進而管理與創造知識，並提供給團隊或組織能夠加以運用。

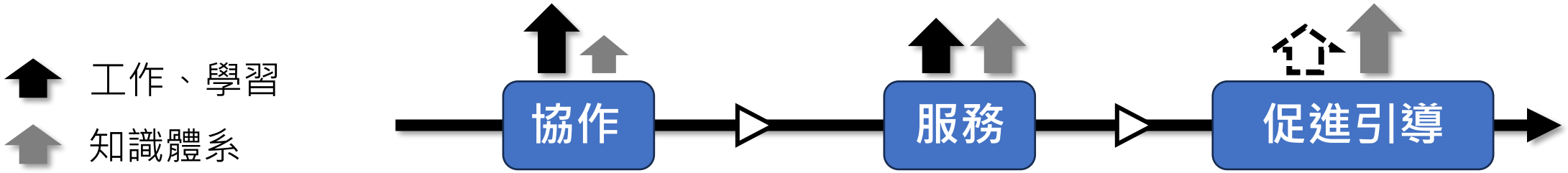
團隊間的互動行為

Access Personal Interface

- 協作
- 服務
- 促進引導

工程師的職業生涯

存取個人介面



工程師要將開發專案時所習得的隱性知識轉化出來成為顯性知識，
進而管理與創造知識，並提供給團隊或組織能夠加以運用。