從 Agentic Software Engineering 出發

115 資工專題 · 2026/07/07 Meeting

規格 → 開發 → 維運 · 財務報帳作為第一個實驗場

起點不是「做一個聊天機器人」

真實的報帳經驗先暴露了兩個問題：

- **名單沒登記**：公文裡有名字，但系統只認登記名單，最後仍被退件。
- **機票艙等對不上**：文件與系統條件不一致，案件退回重做，最後用計畫結餘款補差額。

共同點不是「不知道規定」，而是：

> **制度裡有很多只有資深承辦人知道的條件，新人只能一個一個踩過去。**

當時的企圖：重新設計「規格 → 開發 → 維運」

角色	當時構想
PM	把需求整理成可追蹤的規格
Architect	質疑規格、拆模組、判斷影響
Engineer	交給 AI 實作，不要求人自己下去寫每一段 code
QA	用測試驗收 AI 交付的結果

每次改動都要能回答：

為什麼改？影響哪裡？誰同意？怎麼驗收？

兩層架構：先養規格，再讓 Agent 維護 Agent

META layer

使用者 / 老師提出需求 → Agent 持續整理規格 → Architect 審核 → 人確認

↓

TARGET layer

AI 開發 → 測試 → 上線 → 收到新需求 / bug → 回到 META layer

> **系統不是生成一次就結束，而是要能持續演進。**

人的角色：把關，而不是親手完成所有工作

當時想像的工作方式：

1. Agent 先提出一個可執行方案，不只丟 A/B 選項。
2. 人跟 Architect 來回討論，確認方案真的合理。
3. 規格在討論過程中同步更新。
4. 確認後才觸發開發與測試。

這個方向借鏡 MetaGPT / MGX 類型的 multi-agent 軟體工程流程，但希望補上：

> **一次性生成 → 持續性的版本、審核與維護迴圈。**

老師這次留下的核心問題

- **Challenge 在哪裡？** 特別是使用者回饋怎麼真的進入系統。
- 開發順序選擇：不是先把系統做完再補 chatbot，而是先讓需求收集與規格形成機制站起來。
- 建議兩人一組，AI 當 partner；**規格書要能讓工作真正交接。**

> **這一場還沒有定案做「經費核銷預檢」；下一步是把通用構想壓到一個足夠具體、可驗證的問題上。**