Process model: Plan-driven: 此方法是事先即規劃好每一步驟的輸出以及輸入，每一步的輸出即是下一步的輸入。每一步都會進行詳細的規劃並且擁有良好的順序。適用於目標明確、結果明確、或是擁有許多先前經驗能夠參考的項目。Incremental：此方法是將專案分成許多部分稱為增量，在不同的時間點開啟某增量的開發工作。每一增量內部採用 plan-driven 方式。但最終產品則是透過逐步將這些增量組裝再一起和獲得最終產品。每一增量都是單一可操作的組件，可先進行測試。因此此方法兼具迭代以及 plan-driven 的優點。Lean 精實開發: 強調最大化效益並最小化成本，提高流程效益、減少浪費。其通常都會在開發正式開始之前就有一個詳細的計畫規劃生產流程。Agile: 敏捷開發則強調迭代開發並將生產過程分為多個部分，以應對快速變化的使用者需求。這兩者最大不同是其進行的方式，精實開發流程通常是線性的，確保每一步都能夠最大程度地減少浪費。敏捷開發則會透過 sprint plan 的方式將生產過程分個數個部分，每一部分完成時都會擁有具體的軟體功能，並提供相當的彈性可供修改。formal software inspections 在敏捷開發過程中並不常見的原因主要是由於敏捷方法的特性和理念與 formal software inspections 存在一定的衝突。敏捷開發強調的是快速而有迭代的成品交付，而 formal software inspections 則是對軟體進行全面而完整的功能測試。這會使得開發時間被延長許多。但是在敏捷開發也有許多作法來達到類似 formal software inspections 的效果。例如 unit test，unit test 透過較小範圍的程式碼測試來確保在小範圍內的程式碼能夠正常運行。軟體工程師甚至可以選擇 Test-Driven Development，意即先編寫測試後再撰寫程式碼。

Process Improvement -> Capability Maturity Model (CMM): Framework representing a path of improvements recommended for software organisations. 共有五個階段。1. Initial: 過程比較無序，進度、預算、功能、質量都不可預測。此種等級的團隊不具備穩定的開發能力，在遇到問題時通常放棄原先計畫並專注在編碼和測試。2. Repeatable: 擁有基本的專案管理流程，開發目標明確。可追蹤成本、進度與功能完成度。擁有可接受的 discipline(紀律)在相似度高上的不同專案重複這些過程。3. Defined: 相較於 repeatable 等級來說擁有了標準化的流程。同時也開發出特有的流程來面對項目中的特殊情況。並有能力可以審視並優化工作流程。4. Managed: 可將成本、流程、質量用量化數字表示。其開發的項目的績效變的是可預測的。在開發流程中持續對團隊進行監控並在必要時採取相關措施。並開始有相關的資料庫用以紀錄和分析有關軟體開發的數據。5. Optimising: 擁有最成熟的開發品質，開始嘗試創新的理念以及技術。有能力解決並預防可能發生的錯誤。可完成以下項目代表擁有第二級能力: requirements management, software project planning, software project tracking and oversight(追蹤與監督), software subcontract management(分包管理), certain level of software quality, software configuration management(配置管理).可完成以下項目代表擁有第三級能力: organisation process focus(可點出流程重點), organisation process definition, training program, integrated software management(整合的軟體管理), software product engineering, intergroup coordination(組織內協調), peer reviews 可完成以下項目代表擁有第四級能力: quantitative process management(量化流程管理), software quality management 可完成以下項目代表擁有第五級能力: defect prevention, technology change

Modelling Concepts(User story, use case): User story: 用戶故事通常是簡短的描述，通常用一兩句話來說明需求。強調從用戶或利益相關者的角度來描述需求，通常用格式（As a [user role], I want [feature], so that [benefit]）。用戶故事較為簡單，容易修改和擴展，適合敏捷開發方法中的迭代和增量開發。著重於描述用戶的目標和意圖，而不是具體的操作過程。優點: 促進與用戶的頻繁交流和反饋。便於快速應對需求變更和優先級調整。缺點: 可能缺乏詳細的上下文和技術細節，需要進一步分解和細化。適合小範圍需求，不太適合複雜系統的全面描述。Use case: 用例提供詳細的需求描述，包括系統的功能、交互流程和邊界條件。用例描述系統如何與外部角色（如用戶或其他系統）進行交互，強調系統的行為和功能。用例通常有結構化的描述，包括用例名稱、角色、基本流程、替代流程和前置條件等。用例試圖完整地描述一個功能的所有可能情況，包括正常流程和異常處理。優點: 提供詳細和結構化的需求描述，有助於理解複雜的系統行為。適合用於全面和正式的需求規格說明，有助於設計和測試階段。缺點: 可能過於詳細和複雜，不易於快速變更。編寫和維護成本較高，特別是對於大型和複雜的系統。Product centred model: Focus on features to be delivered - expect users will use features to complete tasks. User-Centred model: Focus on anticipated usage - what do users need to accomplish. Reveal necessary functionality, assists with prioritization. Modelling limitation: Models are never perfect §missing details from problem domain §added constraints not from problem domain

有三個主要的文件提及 SE standards, 其分別是 ISO/IEC 12207:2017, ISO/IEC/IEEE 15288:2015, ISO/IEC/IEEE 15289:2017。總地來說，follow 這些軟體工程標準可以讓我們理解軟體的生命週期，以便對其進行 modeling。另外也可以更好地管理、控制並改善開發軟體的流程。

Elicitation techniques: Interview: 面試，是最正統最常見的需求獲取技術。可以事先準備好問題(尤其是開放式問題)主要用於獲取各式各樣來自使用者的期待與需求。面試者也可以準備一些數據輔助面試問題的進行。也可以事先準備團隊的一些想法，訊問使用者對他們的看法以及建議。Workshop: 工作坊是一種結構化的會議，涉及多個利益相關者，旨在共同討論和定義需求及期望。使用各種活動和工具來促進參與和交流。Focus groups: 相較工作坊規模較小，派出具體代表性的利害關係人，並以獲取廣泛想法的開放性(open-ended)會議。Observation: 觀察是一種直接觀察用戶在實際工作環境中如何使用系統的技術。以收集真實的使用情況和需求。Questionnaires: 問卷調查是一種收集大量人群需求和意見的技術。其結果可以納入面試或是工作坊中討論。以及 independent Elicitation techniques 也就是 System interface analysis , User interface analysis, Document analysis. Limits: Stakeholders&user 描述不全、無法預測到的需求、需求間彼此矛盾等等。

Risk identification, reduction, and analysis: 風險大致上有以下幾種可能: Estimation(軟體各方面的評估錯誤)、Organisation(組織問題)、People(人員問題)、Requirements(需求問題，尤其是改變)、Technology(技術問題)、External(其他外部問題)。

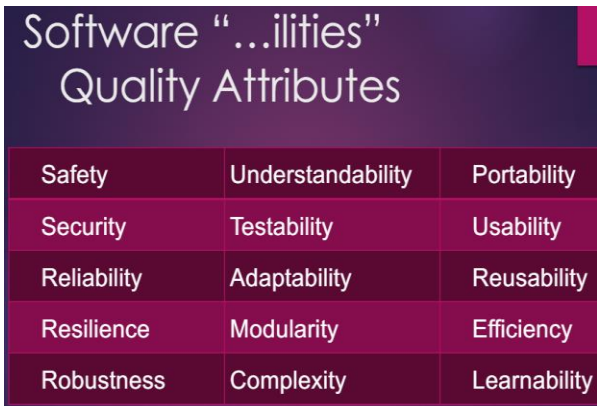
Risk analysis -> 分為嚴重性跟機率。1~4, 5~12, 15~分為三個風險等級。

Level	Value
5	Near Certainty
4	Very Likely
3	Likely (50/50)
2	Unlikely
1	Improbable

Risk reduction: identify 之後自己想。

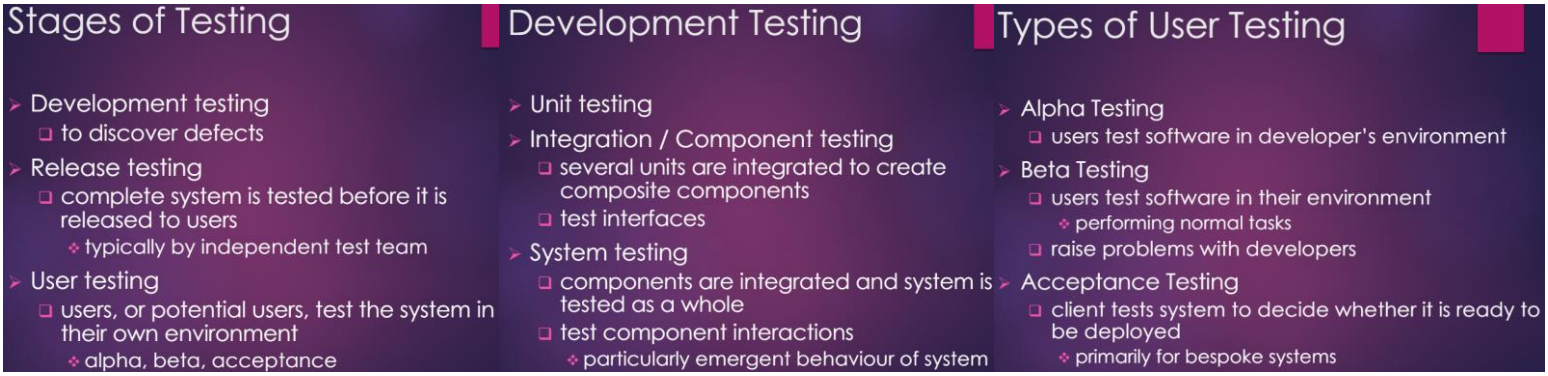
Level	Value	Technical Criteria	Cost Criteria	Schedule Criteria
5	Catastrophic	Product unusable	> $10M or loss of life	Multiple milestone slippage
4	Critical	Product unusable but some work recoverable	> $5M or serious harm	Single milestone slippage
3	Moderate	Functionality degraded but partly usable	> $1M or minor harm	Loss of 1 month in schedule
2	Marginal	Minor loss of functionality or noticeable performance issue	> $100K	Loss of 1 week in schedule
1	Negligible	Trivial loss of functionality or performance	minimal cost	Loss of few days in contingency

Quality Management: Quality, simplistically, means a product should meet its specification.



V&V: Validation & Verification

Why do we test? -> To see if something works? (X. To see if something doesn't work? (O.



Types of User Testing

- Alpha Testing
  - users test software in developer's environment
- Beta Testing
  - users test software in their environment
    - performing normal tasks
  - raise problems with developers
- Acceptance Testing
  - client tests system to decide whether it is ready to be deployed
    - primarily for bespoke systems

Reviews vs. Testing

- Reviews
  - analysis of static system representation to discover problems
    - static verification
  - supplemented by tool-based document and code analysis
- Testing
  - exercise and observe product behaviour
    - dynamic verification
  - execute system with test data

Benefits of Reviews

- Cost benefit from early error detection
  - author receives timely feedback
- Can be applied to untestable products
  - e.g. requirements specification
- During dynamic testing errors can hide other errors
  - static verification means no interaction between errors
- Distributes knowledge

Types of Reviews

- Technical Review
  - conformance to standards
  - achievement of milestones
  - led by team leader
- (Fagan) Inspection
  - peer review with formal process
  - led by independent moderator
  - systematic data collection
  - focus on defect detection
  - process improvement goal
- Structured Walkthrough (Yourdan)
  - less formal than inspection
    - no data collection
    - no preparation
  - led by producer
- Audit
  - external review of artefacts
  - independently managed
  - usually late in process

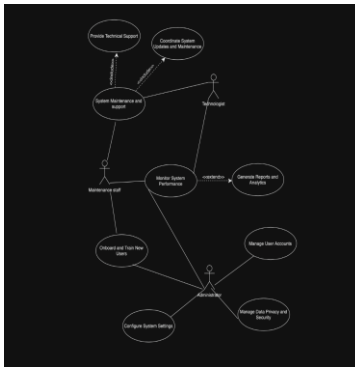
Inspection Principles

- Defined roles for participants
  - requires technical knowledge
  - preparation in advance
- Checklists used to identify defects
  - focus on finding problems
    - not solving problems
- Inspection data is recorded
  - effectiveness of inspection process
  - relationship with product quality

Inspection Process

- Request
  - producer requests inspection of artefact
  - moderator is selected
- Entry
  - moderator checks artefact is ready
- Planning
  - moderator plans inspection meeting
    - e.g. who will participate
- Overview (optional)
  - meeting to distribute documents

Use case diagram



Description of scenario:

Typical Scenario

Actor Stimulus	System Response
1. People for emergency help are ready for any dispatch.	2. They receive dispatch requests from emergency responders.
3. People for emergency help are heading to the patient's location.	4. Emergency responders provide current situations to people for emergency help.
5. People for emergency help arrive at the patient's location and give first aid.	6. Inform social workers of the potential assistance when they arrive at the hospital.
7. Heading to hospital with the patient carried.	8. Social workers preparing for the patients' arrival.
9. Arrival at the hospital.	10. Help patients with the affairs in the hospital with social workers.

Post-Condition(s)	The patient has been sent to the hospital successfully.

Alternative Scenarios

Alternative 1: at step 2

Actor Stimulus	System Response

The communication system is broken unexpectedly	Emergency responders notice people for emergencies have not dispatched yet by monitor, re-inform them by any other tools.

Alternative 2: at step 3

Actor Stimulus	System Response
Insufficient people for emergency help.	Reschedule people for emergency help immediately. Ensure they could deal with the emergencies which have relatively high emergency levels.

User story

1	Tutorial mode	Priority: M	4

As a new player, I want to have a tutorial mode so I can learn the basic mechanics of the game before jumping into multiplayer and getting slammed.

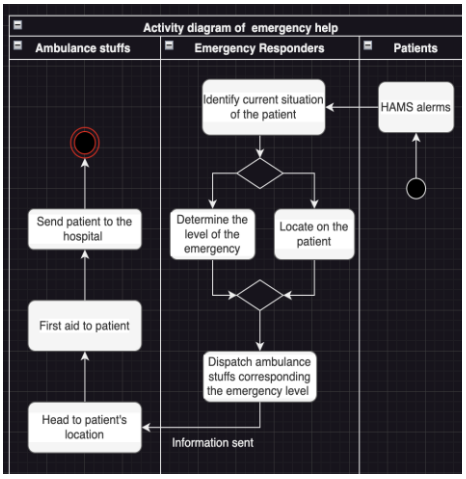
Acceptance Criteria :

- Have a mock keyboard or controller appear on screen and highlight the buttons needed to perform different actions paired with an action
- Add a button for tutorial mode on the game mode screen, which takes the player into an in-depth tutorial to familiarise them with the controls and objective of the game.
- The tutorial will have different stages, from basic controls to an in-game scenario with bots, to help the player get familiar with the game.

Notes

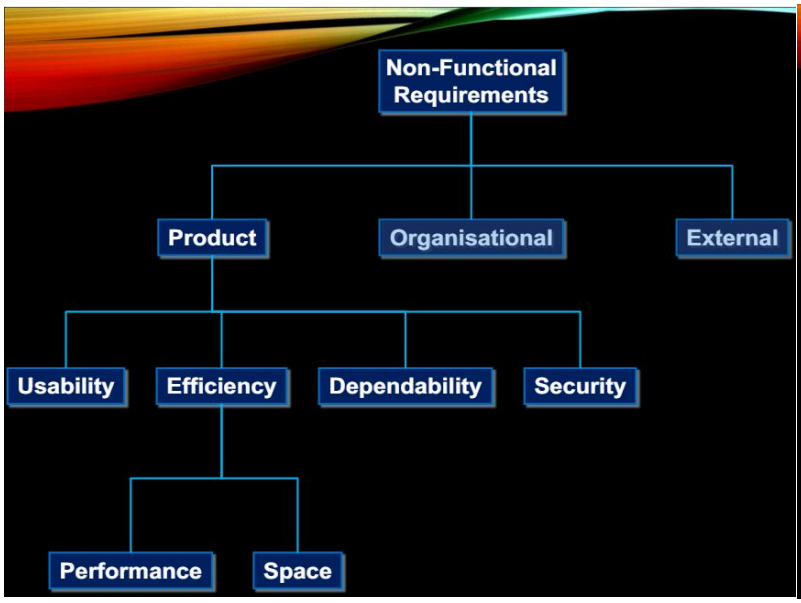
- Trigger actions with right combination of buttons.

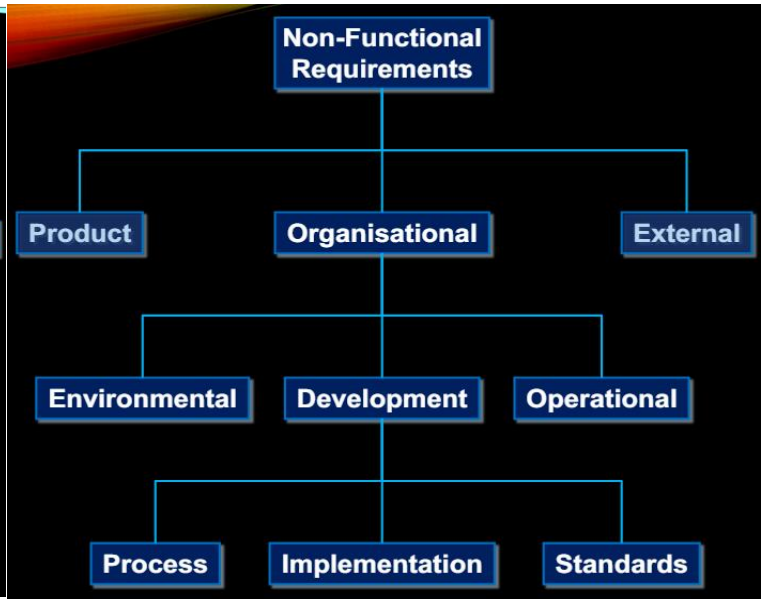
Activity diagram

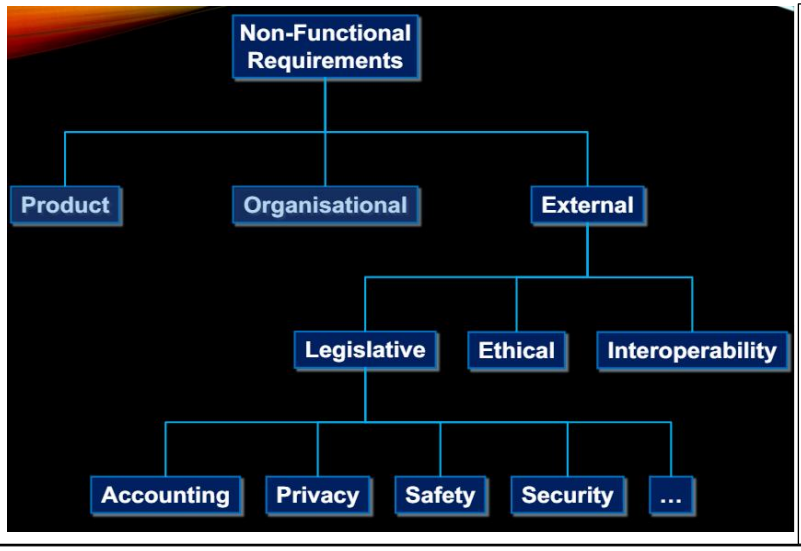


Estimation: Planning Poker, Story Points, Velocity(Team counts number of story points completed at end of each iteration, Probably will complete about the same number in the next iteration)

MoSCoW: - Must have - Should have - Could have - Won't have







NFR: System properties and constraints. Non-functional requirements may affect overall architecture. A single non-functional requirement may: 1. generate a number of related functional requirements that define required system services. 2. restrict functionality of existing functional requirements

Software Size Measures: Types of size measures – syntactic -> e.g. Lines of Code (LOC) – semantic -> e.g. Function Points. Lines of Code(Widely used): 優點:很直觀、很方便計算。缺點: hard to visualise early in a project, hard for clients to understand, does not account for complexity or environmental factors, 需要具備程式語言能力. Function Points: 優點:不依賴程式語言，因此可以廣泛使用。缺點:計算較為主觀(subjective), 計算無法自動化, 較無法適用於新的軟體系統。

FP 內容:**外部輸入（External Inputs, EI）**：用戶提供的數據輸入到系統中的次數，例如輸入表單。**外部輸出（External Outputs, EO）**：系統向用戶提供的數據輸出次數，例如報告、查詢結果。**外部查詢（External Inquiries, EQ）**：用戶對系統數據的查詢次數，包括輸入和輸出數據的組合。**內部邏輯文件（Internal Logical Files, ILF）**：系統內部存儲的邏輯數據文件數量。**外部介面文件（External Interface Files, EIF）**：系統與其他系統共享的數據文件數量。未調整的功能點計算(將以上五個因素根據其難度加總。下左)VAF: 總共有 14 個影響因子(下右)，把他們分數加總之後除以 100 + 0.65 就是 VAF 的數值。通常介於 0.65~1.35 之間。COCOMO 等演算法(algorithmic)類型的成本估算技術的優缺點: 使用像 COCOMO2 這樣的算法成本估算技術的一個優勢是它提供了一個結構化的方法來估算軟件開發的成本和時間。這種結構化的方法基於數學模型，考慮了多個影響因素，包括軟件規模、複雜性、開發團隊的經驗等，使得估算過程更加客觀和準確。然而，使用 COCOMO2 等算法成本估算技術的一個缺點是它可能需要大量的輸入數據和詳細信息，包括軟件規模、成本驅動因子等。這意味著在項目的早期階段，當這些信息還不夠完整或可靠時，估算的準確性可能會受到影響，從而導致估算結果不夠可信或準確。

功能類別	簡單	中等	複雜
外部輸入	3	4	6
外部輸出	4	5	7
外部查詢	3	4	6
內部邏輯文件	7	10	15
外部介面文件	5	7	10

資料通訊, 分散數據處理, 性能, 大量交易, 資料輸入, 使用效率, 儲存容量, 易用性, 資料處理邏輯, 多重使用者, 易變更性, 重新使用, 資料轉換, 裝載和安裝

簡單估算方法: Planning Poker, Wide-Band Delphi: 特點:**匿名性：** Wide-Band Delphi 方法通常是通過匿名進行的，這意味著每個專家都可以自由地表達自己的觀點，而不必受到其他專家的影響。**進化性：** Wide-Band Delphi 方法通常是一個進化的過程，通過多輪的反饋和討論，專家可以不斷地調整和精化自己的估算，從而提高結果的準確性和可信度。**統計分析：**Wide-Band Delphi 方法通常會對專家提供的估算結果進行統計分析，以確定中位數、平均值等結果，並評估專家之間的一致性和差異性。**靈活性：**Wide-Band Delphi 方法通常比較靈活，可以根據實際情況和需求進行調整和定制，例如修改問題描述、增加或減少專家數量等。Story Points, Velocity: 特點:**進度追蹤：** Velocity 可以作為一個指標來追蹤團隊的進度和工作效率。通過比較不同迭代中的 Velocity，團隊可以了解自己的工作速度是否穩定，以及是否有任何改進的空間。**預測能力：** 基於過去幾個迭代的 Velocity，團隊可以使用它來預測未來迭代中可能完成的工作量。這有助於更準確地規劃和管理項目。**質量衡量：** Velocity 不僅僅是工作量的量度，還可以用來衡量工作的質量。一個高 Velocity 可能表明團隊在迭代中完成了許多工作，但如果這些工作沒有達到高質量，那麼 Velocity 本身就不能完全代表成功。**持續改進：** 團隊可以通過定期檢視 Velocity 來識別問題、挑戰和改進機會。如果 Velocity 出現異常波動，團隊可以進一步調查原因並采取相應的行動。

Technology Grid 通常是基於一個矩陣或表格形式展示的，其中行和列分別代表了不同的技術或工具以及它們的特點和功能。下面是一個簡單的示例：

yaml 複製程式碼

	Feature 1	Feature 2	Feature 3
Technology/Tool 1	Yes	No	Partial
Technology/Tool 2	Partial	Yes	Yes
Technology/Tool 3	No	Partial	No

在這個表格中，列代表不同的功能或特點，而行代表不同的技術或工具。每個單元格中的值表示相應技術或工具在特定功能或特點上的支持程度，通常使用 "Yes"、"No" 或 "Partial" 來表示。

Technology Grid 的建立和填充通常包括以下步驟：

1. **確定功能和特點：** 首先，項目團隊需要確定要考慮的功能和特點，這些可能包括功能、性能、可用性、安全性等。
2. **收集信息：** 團隊需要收集有關不同技術或工具的信息，包括它們的功能、特點、優缺點等。這可能需要查閱文檔、參考評論或進行測試。
3. **填充矩陣：** 團隊將收集到的信息填充到 Technology Grid 的相應單元格中，根據每個技術或工具對每個功能或特點的支持情況進行標記。
4. **分析和比較：** 填充完 Technology Grid 後，團隊可以對不同技術或工具在不同功能或特點上的支持情況進行分析和比較，以幫助做出最佳選擇。

親和力估算（Affinity Estimation）是一種敏捷軟件開發方法中常用的估算技術，通常用於估算項目中的工作量、任務或特性的大小或相對複雜度。這種方法的核心概念是通過團隊成員的集體智慧和討論來達成一致，而不是依賴單一專家或數據模型。

如何進行親和力估算？

1. **準備工作：**確定要估算的工作項目或特性，通常這些項目應該被分解成可以輕鬆估算的小單元，例如用例、任務或故事。
2. **定義估算範圍：**明確界定要估算的範圍，確保所有團隊成員對估算的內容和目標有清晰的理解。
3. **使用貼紙進行估算：**將要估算的工作項目寫在貼紙上，然後讓團隊成員集體進行估算。每個成員將自己對工作量或複雜度的估算寫在貼紙上，然後將貼紙貼在白板或大牆上。
4. **討論和調整：**團隊成員對估算結果進行討論，並就差異進行解釋和調整。這個過程通常需要反復進行，直到達成一致或接近一致的結果。
5. **取得共識：**當團隊成員對估算結果達成一致時，確定最終的估算結果並記錄下來，作為後續規劃和執行的依據。

親和力估算的優點和局限性

優點：

- **快速：** 親和力估算通常可以在短時間內完成，適用於敏捷開發中的快速迭代。
- **參與性：** 所有團隊成員都可以參與到估算過程中，從而提高了估算的準確性和可信度。
- **集體智慧：** 通過團隊討論和共享，可以充分發揮集體智慧，使估算結果更加客觀和全面。

局限性：

- **主觀性：** 估算結果可能受到個人主觀意見的影響，特別是當某些團隊成員缺乏相關經驗或知識時。
- **受限於團隊：** 如果團隊成員之間存在權威性或人際關係問題，可能會影響到估算的過程和結果。
- **精確度問題：** 親和力估算往往比較依賴主觀判斷，可能不如基於數據模型的估算方法準確。

T-Shirt Size Estimation 是一種敏捷軟件開發中常用的估算技術，通常用於估算項目中的工作量或任務的大小，而不是具體的時間或成本。它使用 T 恤尺寸（T-Shirt Sizes）作為相對大小的表示，例如 Small、Medium、Large 等，而不是使用具體的時間單位或工作量單位。

T-Shirt Size Estimation 的特點：

1. **相對大小：** T-Shirt Size Estimation 強調的是相對大小而不是具體數值。團隊成員通常根據他們對任務大小的理解和感覺來指定相應的 T 恤尺寸，這使得估算過程更為簡單和直觀。
2. **高層次抽象：** T-Shirt Size Estimation 提供了一種高層次的抽象，使得團隊可以更快速地估算工作量，而不必深入到具體的細節。這有助於在項目計劃和溝通中更有效地使用估算結果。
3. **相對一致性：** 雖然 T-Shirt Size Estimation 不提供具體的數值，但它仍然強調相對一致性。這意味著團隊成員需要就任務的大小達成一致，從而確保估算結果的相對準確性。
4. **用於高層次規劃：** T-Shirt Size Estimation 通常用於高層次的規劃和優先排序，例如項目的發布計劃或Sprint的優先順序。它不是用於具體的時間或成本估算。

如何進行 T-Shirt Size Estimation：

1. **確定任務：** 首先，團隊需要確定要估算的任務或工作項目。
2. **定義 T-Shirt Size：** 接下來，團隊成員將根據他們對任務大小的理解和感覺，指定相應的 T 恤尺寸，例如 Small、Medium、Large 等。
3. **討論和協商：** 團隊成員之間可能存在對任務大小的不同理解，因此可能需要進行討論和協商，以達成一致。
4. **記錄結果：** 將每個任務的 T-Shirt Size 記錄下來，作為後續規劃和優先排序的依據。

Object / Application Points

- Alternative function-related measure to FP
  - suited to DB programming or scripting languages
    - 4GLs
  - easier to estimate than FP
- Object points are *NOT* the same as objects in an OO sense
  - COCOMO 2 uses the term Application Points to avoid confusion
- Can be estimated early in the development process

Application Point Estimation

- Number of separate screens that are displayed
  - simple screens are 1 object point
  - average screens are 2
  - complex screens are 3
- Number of reports that are produced
  - simple reports are 2
  - average reports are 5
  - complex/difficult reports are 8
- Number of 3GL modules that must be developed to supplement the 4GL code
  - 10 object points per module (e.g. a class in Java)

Application Composition Model

- Early "order of magnitude" estimate
  - may be more accurate for 4GL intensive projects
- Based on standard estimates of developer productivity in application points / month
  - considers developer experience and CASE tool use
  - doesn't consider system complexity or developer variability

Estimation Formula

- $PM = \left(NAP \times \left(1 - \left(\%reuse \div 100\right)\right)\right) \div PROD$
  - PM – effort in person-months
  - NAP – number of application points
  - PROD – productivity

Developer Experience and Capability	V Low	Low	Nominal	High	V High
CASE Maturity and Capability	V Low	Low	Nominal	High	V High
PROD (NAP / Month)	4	7	13	25	50

- FP 的替代功能相關措施

§ 適用於資料庫程式設計或腳本語言

◦ 4GL

§ 比 FP 更容易估計

- 物體點與物體內的物體不同。

物件導向的意義

§ COCOMO 2 使用術語「應用點」以避免混淆

- 可以在開發過程的早期進行估計

早期「數量級」估計

§ 對於 4GL 密集型項目可能更準確

- 基於開發人員生產力的標準估計（以應用點/月為單位）

§ 考慮開發人員經驗和 CASE 工具使用

§ 不考慮系統複雜性或開發人員變化性

- 顯示的單獨螢幕的數量

§ 簡單的螢幕是 1 個物件點

§ 平均螢幕數為 2

§ 複雜螢幕有 3

- 產生的報告數量

§ 簡單報告 2

§ 平均報告數為 5

§ 複雜/困難的報告為 8

- 必須開發以補充 4GL 程式碼的 3GL 模組數量

§ 每個模組 10 個物件點（例如 Java 中的類別）