CSIE5142/CSIE4302 Software Engineering

Homework # 1

Due on 10/11/2022S

學號:111598031 姓名:楊佑鴻

1. (10%) Give a brief description about the following terms.
(a) (5%) What is software engineering?
(b) (5%) What is system engineering?

(a)

軟體工程是一種注重於軟體開發各個方面的軟體原則，包括分析用戶需求並進行分析，建構並測試軟體是否滿足客戶的需求。

(b)

系統工程較為關注於以計算機基礎的系統開發的所有方面，如硬體、軟體及程序工程等等。通常專注於如何設計、開發和管理在其生命週期內的複雜系統。

2. (15%) Select one Knowledge Area (KA) in the SWEBOK v3 and briefly summarize 3 breakdown topics at level 1 for that KA. (You may download the
SWEBOK V3 at http://www.computer.org/web/swebok/v3)

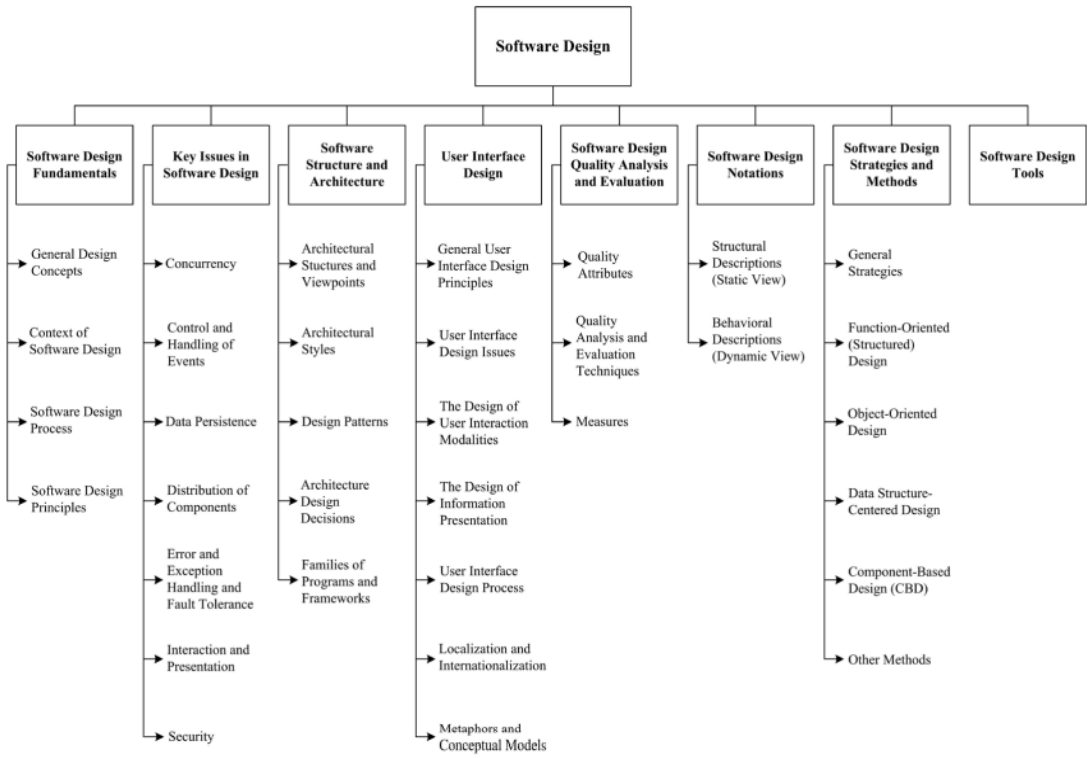


Figure 2.1. Breakdown of Topics for the Software Design KA

- 軟體設計

軟體的「設計」被定義為定義一個系統或元件的結構、元件、介面及其他特質的過程以及其過程的結果。而若是作為過程，軟體設計是軟體工程生命週期的活動，在此過程中軟體需求被很好地分析，為了能提供更細節的軟體內部架構，從而將其作為建構軟體的基礎。軟體設計描述了軟體的架構，也就是軟體如何被分解和組織成為元件，以及這些元件之間的介面。

- 軟體設計基礎

在一般理解下，「設計」這一概念被視為解決問題的一種形式。在理解設計的概念時，進一步理解它的目標、限制、替代方案、表現方式及解決方案等等也是值得深思的。就如同上述的概念，不難看出軟體設計是軟體開發過程中相當重要的部分。

- 軟體設計中的關鍵問題

主要是設計軟體時必須處理的一些關鍵問題，如何分解、組織和包裝軟體元件，如軟體設計之並行設計。並行設計涉及將軟體分解為程序、任務及執行緒。以及處理相關於效率的問題，如原子性、同步性以及排程。

- 軟體結構和架構之建築建構及觀點

描述及記錄了不同的高級層面的軟體設計。這些方面通常被稱作視圖—視圖表是了軟體架構的部分方面，並顯示了軟體系統的特定屬性。
視圖涉及與軟體設計相關的不同問題，例如邏輯視圖（滿足功能需求）與流程視圖（並行問題）vs. 實體視圖（分佈問題） vs. 開發視圖（如何將設計分解為實作單元，並明確表示單元之間的依賴關係）。總之，軟體設計是由設計過程產生的多重方面的產物，通常由相對獨立和正交的視圖所組成。

3. (50%) Process Models
(a) (10%) Describe the characteristics of the plan-driven and agile processes.
(b) (10%) Draw and describe the major activities of the waterfall process.
(c) (10%) Draw and describe the major activities of the Scrum process.
(d) (10%) Draw and describe the major activities of the Unified Process (UP).
(e) (10%) Describe the differences between the plan-driven and agile process
models in terms of their philosophy and applications.

(a)
Plan-driven 是在最初階段預先計畫並規劃所有的流程及活動，按照階段有序地開發應用程序，再根據進度狀況控制流程。

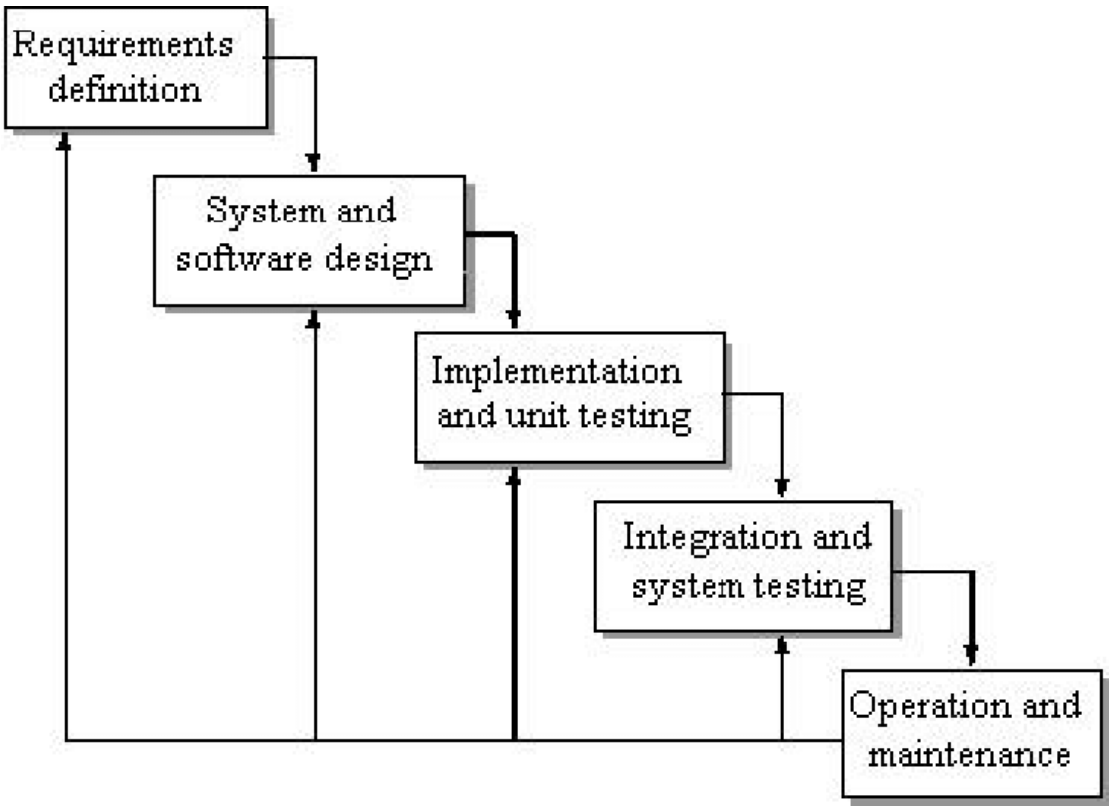
- 專案被分為多個 stages/tasks，並有周期性的規劃。
- 文件相當正式且結構化

- 詳細定義專案中每個角色的職責與責任。
- 因使用者需求改變，做出反應的成本較大。

Agile 主要以增量(incremental)的方式開發，並且能夠根據客戶不斷變化的需求做出反應，從而改變流程。

- 基於軟體開發的迭代(iterative)及增量(incremental)方法
- 透過及早且連續地交付有價值的軟體，能夠提早得到使用者回饋並及時改善。
- 注重於人與人彼此合作多於書面形式的冷溝通，文件合約為輔助。
- 具備靈活性，能夠快速應對需求改變導致的變化。
- 對於使用者需求改變所耗費的成本較低。

(b)



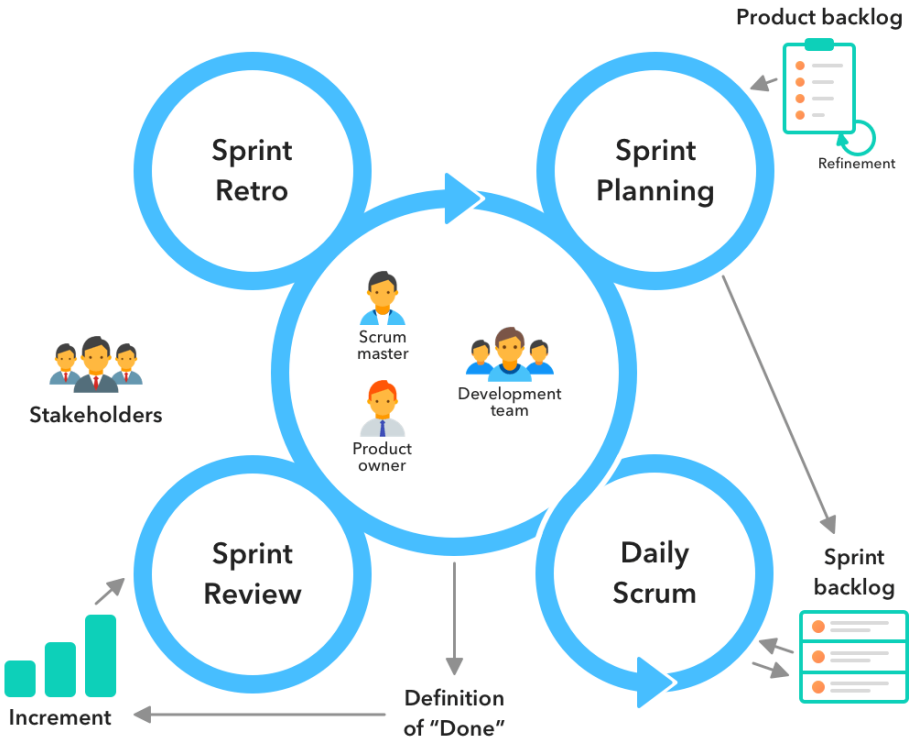
Waterfall process 便是典型的 SDLC 模型。將軟體生命週期規劃為一組階段式的流程，該模型必須在前一個階段完成後才能執行下一個階段，也就是說一個階段的輸出將會是下一個階段的輸入。

Waterfall process 各個階段如下:

1. 需求分析與定義
   在此階段業務與客戶會一起工作，了解客戶的確切要求，系統的服務/限制及目標，並正確的紀錄和寫出系統需求的規範文件。
2. 系統與軟體設計
   根據文件再進一步規劃軟體或硬體的整體架構，並依照需求分配至軟體或硬體系統。

3. 實作與單元測試
   透過設計階段，軟體開發人員所需的訊息皆在文件中，軟體設計能夠順利地實作，並進行以元件為單位做檢驗的單元測試，以確保設計與實作是否符合需求。
4. 整合與系統測試
   將獨立的程式單元或模組進行整合並成為一個完整的系統，並在此階段進行系統測試，以確保系統能夠正常運作。
5. 系統運作與維護
   在這階段客戶會經常性使用產品，若是發生錯誤便快速修正，確保應用程序能夠平穩運行，此階段基本上會維持最久。

(c)



Scrum process 是 Agile process 的典型代表，採用跌代式與增量式的方法可以用有彈性的方式解決複雜的問題，及早地交付具有高價值與高品質的產品。

Scrum process 包含如下四個重要事件:

- Sprint Planning

在此階段，每個 Sprint 開始之前皆會舉辦一次 Sprint planing 的會議。
Scrum 團隊將一起討論並計畫在未來一段時間內要完成的工作範圍，並代辦列表中選擇團隊打算在一次 sprint 完成的項目。
假如團隊認為此次 sprint 無法完成某些 backlog，可能會有將 Backlog 被拆分並回到 backlog 的情況發生。

- Daily Scrum

Scrum 團隊在 Sprint 的期間每日皆會舉辦會議，基本定在同一時間及地點。限定會議於十五分鐘之內，主要表達並回應三個問題: 前一日做了什麼、今日的目標為何以及在項目進行時遇到的障礙。藉由這三個問題來了解整體團隊的工作狀況。

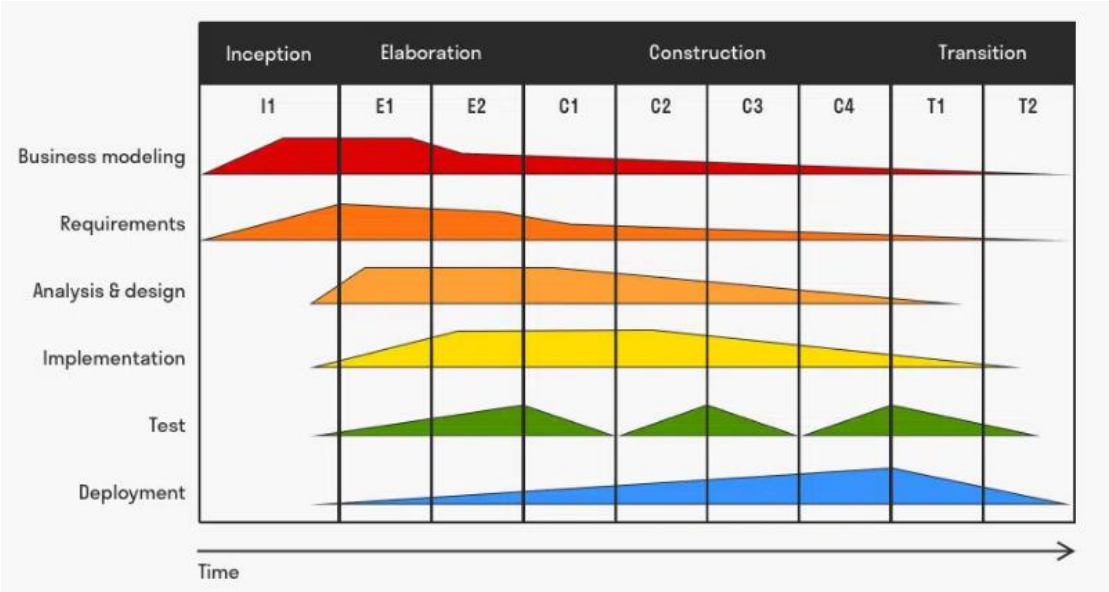
- Sprint Review

在 Sprint Review 的期間，整個團隊會回顧本次 Sprint 完成的工作項目，以及尚未完成的潛在工作。

- Sprint Retrospective

與 Sprint Review 相似，Scrum 團隊將在這個階段反思本次 Sprint 哪個部分做的很好，是否有什麼地方能夠改進。並藉此持續進行改善，調整之後 Sprint 的行動。

(d)



Unified Process 是一種重量級過程，運用跌代式方法針對系統進行開發和細化，具有循環式反饋和適應性能力的軟體工程模型。

跌代分布在四個階段，每個階段由一個或多個跌代組成:

- Inception phase

構思階段，主要在確定項目的基本思路以及其結構。團隊會定期開會已確定項目的必要性、可行性及適用性。

- Elaboration phase

細化階段，在此階段會評估並分析系統的需求及所需要的架構，可以看出這是項目開始漸漸成形的階段。主要的目標是分析產品並為未來的架構規劃基礎。

- Construction phase

建設階段，以前次階段規劃的架構為基礎，軟體系統能夠完整地被建構起來，重點在於開發系統的元件和它的功能，大多數實作便是在這階段開始。

- Transition phase

過度階段，在此階段會將完整的產品轉移給客戶，客戶在這階段會頻繁使用系統，因此幾乎會出現需要對系統進行維護的問題。這階段便是確保產品能夠正常且平穩的進行轉移。

每個階段及跌代都由一組已定義的活動所組成:

- Business Modeling
  描述使用業務案例，針對業務程序進行建模。
- Requirements
  針對系統的需求進行分析與考量，開發者與使用者應對系統需求達到共識。雖與其他模型的需求規則相當類似，但還包含了撰寫 use cases 及確定非功能性需求。
- Analysis & design
  分析與設計涵蓋設計的各個方面，包含了整體架構分析與設計、元件、服務及模組相關之設計、介面設計等等。
- Implementation
  主要為實作並建構整體系統。
- Test
  涉及了測試活動以確保此次跌代中發布軟體的品質。測試其中包含如測試計畫、開發測試場景 alpha 及 beta 測試，回歸測試以及驗收測試。
- Deployment
  對已開發系統進行規劃及部屬的部屬活動。

(e)

	Plan-driven	Agile
相同點	兩者皆是以有紀律的方法進行軟體開發 提供各自的流程、工具及技術的開發方法 沒有絕對的好壞，只有彼此各自適合使用的情況	
相異點	強調有計劃地進行開發	強調富有靈活的方式開發
	以階段式開發為主	以迭代及增量式開發為主
	注重於詳盡的文件及專業的文件格式	注重於人與人之間的互動，文件多用於輔助
	階段式開發之下，需求改變所耗費的成本較大	迭代及增量式開發之下，需求改變成本較低
	強調初期便完成所有需求預測	強調精簡、延遲決策

4. (15%) CMMI
(a) (5%) What is CMMI? What is it used for?
(b) (5%) What is CMMI staged representation?
(c) (5%) Explain the advantages of implementing CMMI.

(a)

CMMI 全名為 Capability Maturity Model Integrated，即為「能力成熟度模型整合」。是一個流程改進的方法，利用控制、量測、改善(control, measure, and improve)等等循序漸進的方式，達到軟體流程改善的一個框架。

CMMI 用於啟動橫貫一個專案、一個部門或一個完整組織的流程改進，提高企業管理的能力。

(b)

CMMI 的階段式表示法可以分為五個階段，每個階段皆提供了一系列的改進目標，且每個階段皆作為下一個階段的基礎。

- ML 1 Initial — 初始階段，企業組織雖然具有開發軟體產品的能力，但是掌控專案時程、成本、流程的能力不佳，經常超過專案預算與時程，因此品質較低且風險較高。

- ML 2 Managed — 管理階段，具備初始階段的能力，且對於專案的需求、流程、產出物具有管理能力，能在特定時間點(如專案里程碑到達時)交付產出物。但流程則可能因專案而異，仍未加規範，品質相較於初始階段高。

- ML 3 Defined — 定義階段，企業組織具備一套標準、適當的管理措施，而且能夠根據企業的特殊狀況及標準流程，從而將這套管理體系與流程加以制度化。此時產品具備中等品質且有中等風險。

- ML 4 Quantitatively managed — 量化管理階段，具備以上階段的能力，因此具備足夠的管理開發能力。以數量方式控制並統計分析，進一步定義流程變異的原因並加以修正，並利用各種測量指標進行量化。

- ML 5 Optimizing — 最佳化階段，具有以上階段的能力以及量化後的流程數據，基於回饋的數據加以分析，因此能夠持續改進開發流程，並研擬創新的技術對整體品質進行最佳化。

(c)

使用 CMMI 能夠具備以下優點:

- 目標性
  透過實施 CMMI 方法，企業能夠根據目前所處的階段，明確地知道下一階段的目標是什麼，並進一步對自身進行調整及改善。

- 一致性
  實行 CMMI 方法循序實現每一階段的目標，對於開發流程及管理方法有增量式的理解，並依此將其流程化，不只如此，更進一步統計並分析相關數據，加以調整與改善，一步步地達成整體一致性。
- 持續改善
  對於成熟的軟體公司而言，最佳化永遠不會有終點。這也便是第五階段最重要的目的之一「持續改進的能力」。不僅推動每個專案的流程改進，更推進組織本身的進化。
- 高品質
  企業組織向每一階段前進，逐步改善開發流程與團隊管理的方式，能夠確保更好的品質並且降低可能的潛在風險。

5. (10%) Theory and Practice
(a) (10%) What factors or selection criteria should be considered when choosing a process model for a software project?

我認為不同類型的軟體應用程式有各自適合的軟體過程模型(Software process model)，因此根據開發的軟體系統選擇軟體過程模型是很重要的。以專案的方面來看，應根據專案規模大小、複雜度、時間等等之限制做考量，以人的方面來看，應根據客戶對軟體的需求、投入成本之多寡而決定。以上諸多因數皆會大幅度地影響軟體系統的各個方面。但即使進行多方面評估，所採用的模組並不代表萬無一失，也會有其因應的挑戰及風險。

以 Plain-driven 及 Agile process 對應四大問題類型為案例，若是 Simple problems 類型的專案，複雜度相對較低，耗時短，需求在開發期間不會有太大的變化，依據這些因素可得知套用 Plain-driven 模式相對合適。但並不代表 Agile process 不能用於 Simple problems，只是 Agile process 強調於迭代、增量的方式來完成專案，套用 Agile process 消耗相對較多的時間及成本。Complicated problems 類型的專案介於兩者中間，複雜度及耗時程度居中，且需求並非難以斷言，卻須經過時間才能確定。此時便應依據實際情況做取捨，因去考量 Agile process 重複進行迭代所消耗的成本及精力是否值得，若 Plain-driven 所耗成本較低，那就不必猶豫。而當問題轉移到 Complex problems 類型時，因為 Complex problems 無法在事前針對需求做預測，因此很難得知因果關係，所以沒有辦法事前擬定解決方案，此時 Plain-driven 的特性就與其相衝突了，採用 Agile Process 能夠因易變之需求適時調整流程及目標，以迭代式方式逐步完成任務會是更好的選擇。Chaotic problems 類型的專案就如形容，一切完全是未知的情況及領域，Plain-driven 的特性完全無法發揮其作用，沒有經驗可循。Agile process 在如此狀況雖也未必能妥善處理問題，但是其特性能適時發揮作用，不會讓專案停滯不前，以迭代式行動減少損失。