CSIE5142/CSIE4302 Software Engineering

Homework # 2
Due on 11/08/2022
學號:111598031 姓名:楊佑鴻

1. (30%) Software Process.
   (a) (5%) What would be the practical problems when using Scrum for large, long-lifetime systems?
   (b) (15%) Specify the possible factors that can influence the choice of the approach, plan-driven or agile method, for software system development. Please discuss the factors from the **System**, **Organization**, and **Team** perspectives.
   (c) (5%) Discuss why scaling agile methods for large systems is difficult?
   (d) (5%) What are the key characteristics of **Multi-team Scrum**?

(a)

針對小型及大型專案，採用Scrum方法被證實是成功的。但是針對大型且有較長生命週期的系統，採用Scrum卻會遇到不同方面的問題，包含大型系統為眾多系統的集合、Brownfield 開發、不同的利益相關者(Stakeholders) 、長期採購時間、系統配置及規範限制等等問題。

Scrum較為適合用於新軟體開發，而不適合軟體維護。然而，有較長生命週期的大型系統更加重視於部屬後的維護階段，因此大型公司大部分的軟體成本都用於維護該公司現有的軟體系統，而不是將重點放在新系統的開發。

大型系統基本上都是Brownfield systems，意即這些系統包含且與許多系統連動，而許多的系統需求都與這種連動有關，因此並不適用Scrum所強調的靈活性(Flexibility)與增量式開發(Incremental development)的特性。

大型且長生命週期的系統有著很長的採購與開發時間，而 Scrum所強調的Cross-functional team是由一群具備不同專業知識的人所組成，長生命週期的系統將使得團隊成員無法隨意離開團隊，這對於正在持續經營的公司來說並不實際，除非有制定開發後續的團隊方案，否則很難維持團隊的持續運作。

Scrum開發的非正式性可能與公司常用的合約定義所使用的法律條款有所衝突。敏捷開發所強調的是互動為主、合約為輔的觀念，而公司的運作方式可能是以合約為重 ，因此在觀念上也有所衝突。

Scrum方法是為了小型協作團隊所設計的，但現在很多的軟體開發模式都涉及全球分布式團隊。大型系統通常是各個不同單獨系統的集合，由多個獨立的團隊獨自開發部分系統，並且這些團隊都是分布於不同的地點，甚至是不同的時區，因此很難達到Scrum所提倡的團隊開發。

(b)

從系統的角度來看，採用plan-driven 或 agile method 取決於開發系統的類型、生命週期、及規模大小與其規定。

以系統類型的方面來說，針對不需要前期做好詳盡需求分析的系統，可以考慮採用 agile method。Agile method 可以隨時應對客戶的需求，連續交付有價值的軟體，得到客戶的回饋，並根據客戶的回饋改善，更能貼近客戶的需求。但若是該系統類型在初期就必須要有詳盡的規格與設計方式(如具有複雜反應時間需求的即時系統)，那麼採取plan-driven的方法才適合該系統類型。以系統的生命週期來看，有較長生命週期的系統可能需要更多且更詳盡的設計文件，以方便系統開發者將設計意圖紀錄下來，以便與後續的團隊做溝通，因此使用plan-driven的方法會較為合適。系統的規範限制也是影響的原因之一，倘若該系統是需要受到外部監管機關批准的話，那麼可能需要就需要提供更加詳盡的文件，作為系統安全案例的一部分，因此 plan-driven 的方法才能符合該限制。

從組織的角度來看，組織的合約、交付方式及文化是影響採用plan-driven 或 agile method的原因。
agile method缺乏事前詳盡的文件及需求分析，這可能對部分組織造成不便。缺乏文件可能導致合約內容的缺失，進而無法完全保障雙方的權益。
對於許多大型公司的軟體開發已有自己一套的開發模式及程序，特別是傳統的軟體公司，基本上都是以plan-driven的開發模式為基礎。並且，Plan-driven的開發方式在初期便完成設計文件與需求分析，儘管開發期間變數太多，不大可能完全照著計畫進行，但是plan-driven提供了一種虛假的安全感，能使得客戶感到安心，組織也能對客戶有所交代。
並且，應要考慮以agile的增量式方法進行開發，客戶是否能夠及時提供回饋? 若是不能即時回饋並改善需求，那失去使用agile method的優勢 — 及早的客戶回饋。

而從團隊的角度來看，團隊成員的能力、位置分布與職能權限會是影響選擇plan-driven 或 agile method的要素。
基於 plan-driven 所組成的團隊成員只需要按照詳細的設計文件進行實作即可。相較於此，因為 agile method所採用的是cross-functional team的組成，採用 agile method 的開發模式的團隊可能須具備更高階且更專業，甚至是不同領域的專業技能。而基於 plan-driven 所組成的團隊並沒有地區方面的限制，因為前期已經做好分析，照設計文件實作即可。而 agile method則不同，agile method以小型的同地團隊為基礎，以便組員能夠順利溝通。因此若是不能滿足同地團體的條件，那麼該團隊就並不適合agile method的開發模式。團隊所擁有的職能權限也是考慮的因素之一，plan-driven 的團隊並不需要完全的團隊自主權，便可正常運作。而 agile method的團隊強調團隊的自主性，團隊要能夠進行sprint review，選擇product backlog items，自我調節與掌控進度。

(c)

當要擴增敏捷方法，維持敏捷開發的基礎是最重要的。我們應該要考量是否能夠達成靈活規劃、頻繁地釋出系統、持續整合、測試導向開發及良好的團隊溝通。
而當我們要為了要能開發大型系統而擴增敏捷方法時，應要注意以下重點:

應對需求工程的完整增量式方法是不存在的。
在開發大型系統時不能只有一位 Product owner 或 Customer representative。
對於大型系統開發，不可能只重視於系統的程式碼，而會需要更多的前期設計與文件。
必須良好的設計與使用跨團隊之溝通方式(Cross-team communication)。
當有多個程序需要被整合為系統時，持續整合實際上是很難實現的。但是卻必須保持頻繁的系統建構與常態的系統發布。

(d)

Multi-team Scrum的關鍵特色包含以下四種：

- Role replication 角色複製
  每個團隊皆擁有一位負責他們工作組合的Product Owner及一位ScrumMaster，而整個專案可能有一位Chief Product Owner和Scrum Master。
- Product architects 產品架構
  每個團隊會選擇一個產品架構師，這些架構師會協助設計及發展整個系統架構。
- Release alignment 釋出對齊
  每個團隊的產品發布日期是一致的，以便可以在發布日期能夠展示可驗證、完整的系統。
- Scrum of Scrums
  每日都有daily Scrum，每個團隊的代表在開會時討論目前進度及計畫要完成的工作。

2. (20%) Consider the requirements engineering process.
   (a) (5%) Describe the major **requirements engineering** activities.
   (b) (10%) Draw an activity diagram to explain **the process of requirements change management**.
   (c) (5%) Describe the concept of **traceability** in requirements management. How can the traceability of requirements be used in the software development process?

(a)

需求工程依據應用領域(Application domain)、涉及之人員與開發需求的組織而有所差異。但是普遍都會包含以下活動:

- Requirements elicitation and analysis

需求獲取與分析，也被稱作需求發現。通常涉及科技人員與客戶共同合作以找到應用領域、系統應該提供什麼服務、系統的操作限制及硬體的限制等等。而在此活動可能涉及終端使用者、管理員、維護工程師、領域專家、公會等等，這些人通常被稱為利益相關者(Stakeholders)。

- Requirement specification 需求規格

將使用者與系統的需求紀錄到需求文件中的過程。

End-user及沒有相關技術背景的客戶需要能了解使用者需求。
系統需求的需求更加詳細並且包含更多的技術資訊。這些要求可能是系統開發合約的一部分。

- Requirements validation 需求驗證

重視於證實需求定義是符合客戶真正需要的系統，需求認知的錯誤將會造成鉅額花費，這也是為何需求驗證相當重要。

- Requirements management 需求管理

需求管理是在需求工程及系統開發期間，負責管理需求變更的程序。新需求會隨著系統的開發與部屬使用隨之出現。需要持續追蹤個人需求及維護相關需求之間的關係。

而在實際開發中，需求工程其實是一種迭代活動，其中的過程是交錯發生的。

(b)

需求變更管理(Requirements change management)可以分為三個不同的階段:

- Problem analysis and change specification 問題分析與規格變更

在這個階段，問題或者變更提案被仔細分析以檢驗是否是有效的提案，而分析的結果會回饋給變更提議者，收到回饋的變更請求者可能會以此提出更加具體的需求變更提案，又或者決定撤回請求。

- Change analysis and costing 變更分析與成本計算

使用可追蹤性資訊及系統要求的一般知識以評估這個提議更改的效果。一旦分析完成，將決定是否繼續實行需求變更。

- Change implementation 變更實施

修改需求文件，在必要時系統設計及實施。理想情況上，需求文件應要被合理組織，從而能輕易實施變更。

(c)

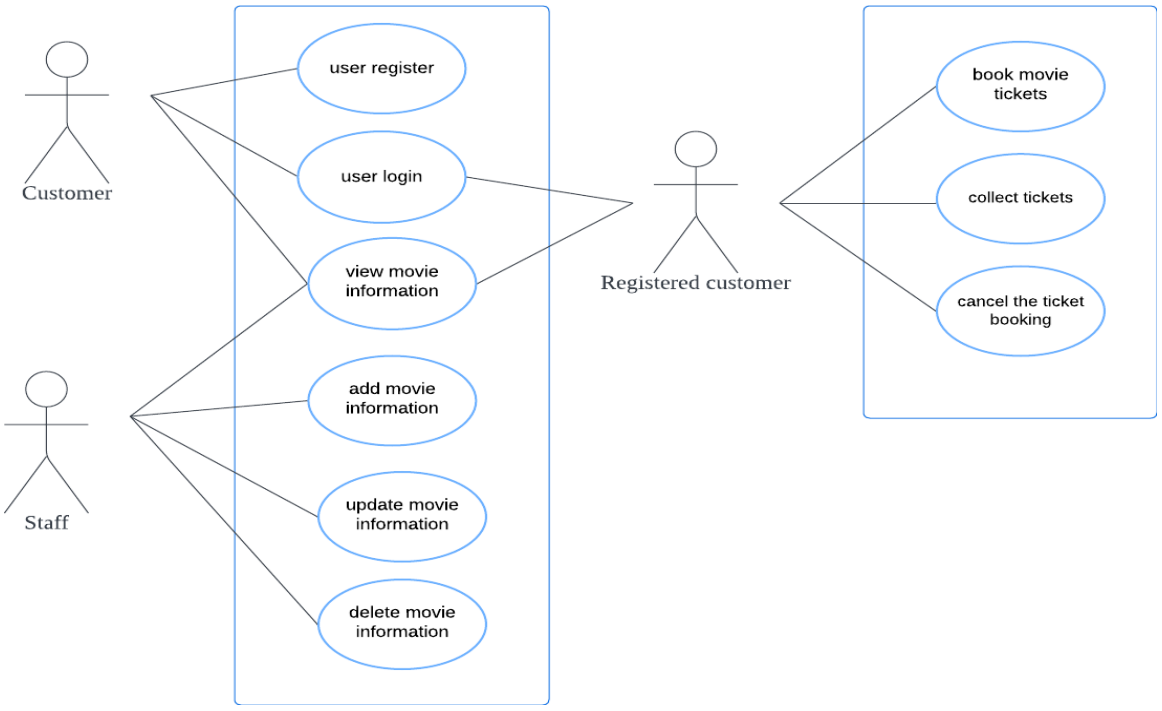
需求可追蹤性是指在產品開發週期對需求進行追蹤的能力，提供針對每個需求的前向、後向追蹤能力(追蹤來源、開發過程及對應的規格、到最終的部屬與使用，以及過程中每一個週期中相關的細化及迭代)。利用可追蹤性可以加速開發專案，越完整的可追蹤性越能減少軟體中的錯誤。

需求可追蹤性能夠幫助專案團隊克服在需求管理中的挑戰，例如在需求改變時可透過追蹤的資訊，進而找到有關及受其影響的產物，確認產物的狀況是否正確。在設計階段時，透過可追蹤性的特性可以確保需求中沒有遺漏的項目，確認所有需求都有實現。在測試階段時連結需求、原始碼、測試案例及測試結果後，若是測試失敗的話可以識別可能是哪一段程式影響的，甚至可能可以找到多餘的測試案例，進而刪減不必要的測試案例。

3. (50%) You have been appointed to develop a **Movie Ticket Booking System** that allows customer to view movie information including description, theater, and show time, to book tickets, to cancel tickets, and to collect tickets. It also allows staffs to add/update/delete/view (i.e., CRUD) movie information. For booking tickets, customers will first login as a registered customer. The registered customer then can select the movie, theater and the show time. To book tickets, the system will retrieve and show the current available seats from the reservation database for the selected movie, theater and show time. The customer then can choose his/her favorite seats and make a payment to reserve the tickets. The system will return a reservation number to the customer if the ticket booking is successful. The customer can collect the tickets by showing the reservation number to the staffs in the cinema. The customers may cancel the ticket booking 30 minutes before the show time of the movie and the system will refund the payment back to the credit card (or account) of the customer.

   (a) (10%) Draw the use case diagram for the system. The use case diagram must have at least two actors, such as customer and staff.
   (b) (10%) Write the user case for booking a movie ticket with the use case template. (Please give detailed description of the use case.)
   (c) (15%) Identified the possible domain classes, such as Customer, Movie, Theater, Showtime, Seat, Ticket, Payment, Reservation, and draw the UML class diagram for the system. Please give essential attributes and methods for each class and specify their visibility. Show appropriate relationships between the classes and specify the multiplicity for each relationship.
   (d) (15%) Draw the UML sequence diagrams based on your class diagram for the scenarios 1) a customer successfully books a movie ticket

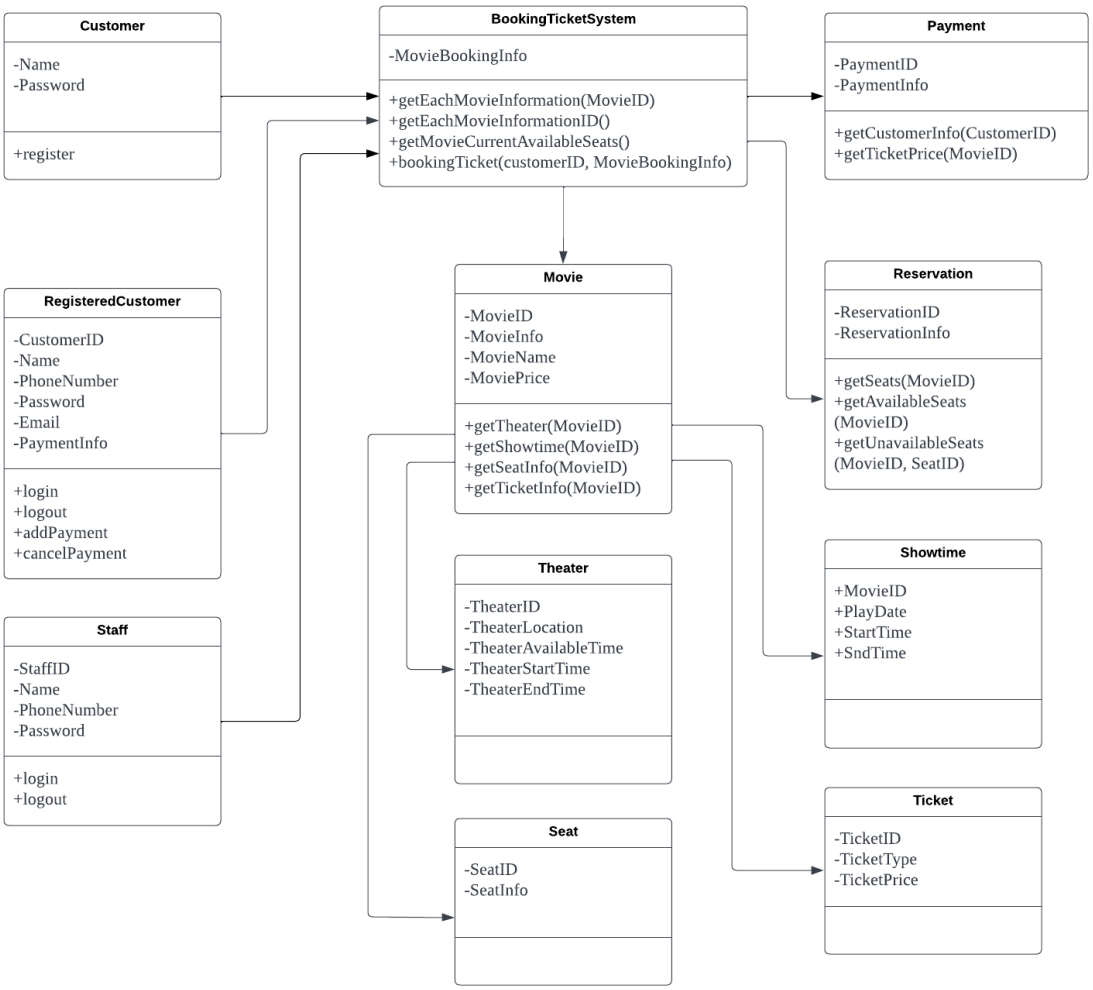
(a)


Customer
Staff
user register
user login
view movie information
add movie information
update movie information
delete movie information
Registered customer
book movie tickets
collect tickets
cancel the ticket booking

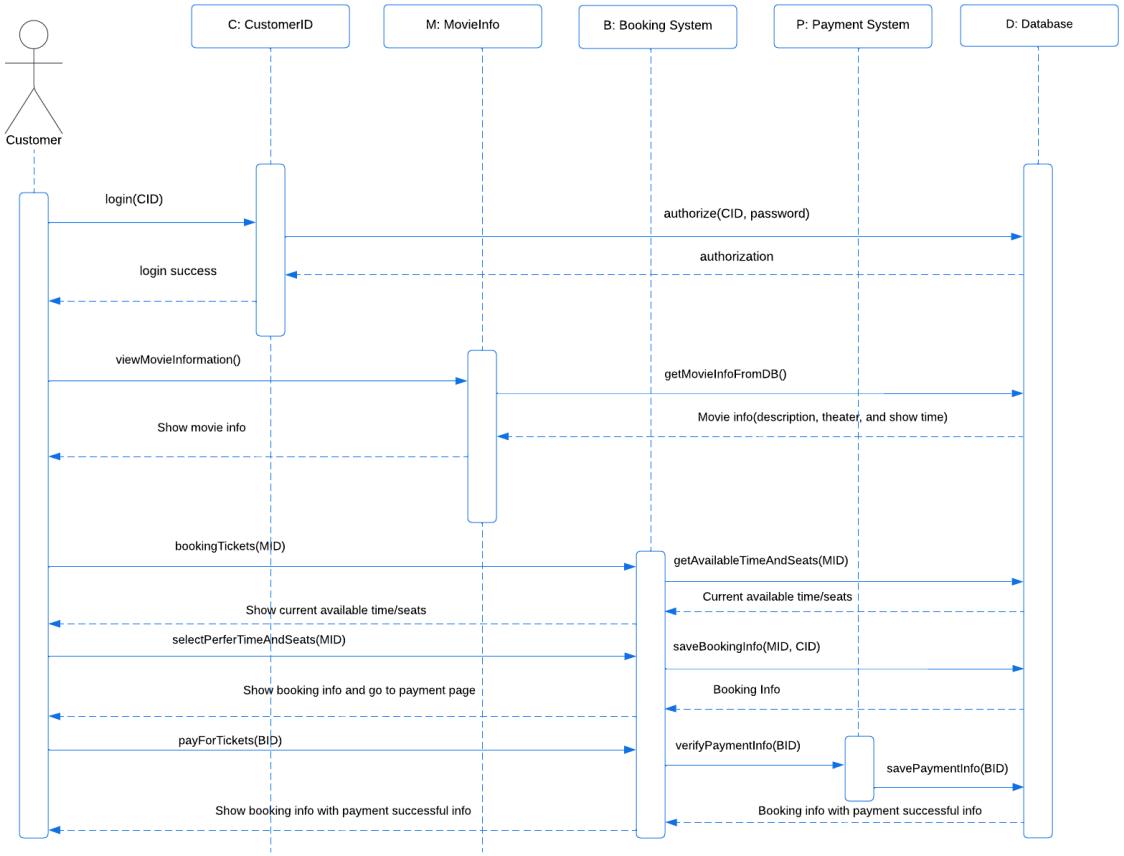
(b)

ID	1
Title	Booking a movie ticket
Description	已登入會員訂購電影票
Primary Actor	已登入會員
Preconditions	需註冊且已登入會員
Postconditions	已登入會員收到訂票資訊及付款通知
Main Success Scenario	1. 登入 Movie Ticket Booking System 2. 選擇想要看的電影 3. 點擊想要看的電影 4. 點選訂票功能 5. 查看目前可訂購的場次及位置 6. 選擇喜歡的場次及位置 7. 選擇購票的付款方式 8. 收到付款完成的通知 9. 支付系統保留會員購票資訊，產生購票訂單。
Fail Scenario	會員付款失敗，將顯示付款失敗的頁面，並且通知該會員訂票失敗。
Frequency of Use	一天最高訂票次數可達8000次
Status	尚未開始
Owner	訂票系統
Priority	1

(c)


Customer
-Name
-Password
+register
BookingTicketSystem
-MovieBookingInfo
+getEachMovieInformation(MovieID)
+getEachMovieInformationID()
+getMovieCurrentAvailableSeats()
+bookingTicket(customerID, MovieBookingInfo)
Payment
-PaymentID
-PaymentInfo
+getCustomerInfo(CustomerID)
+getTicketPrice(MovieID)
RegisteredCustomer
-CustomerID
-Name
-PhoneNumber
-Password
-Email
-PaymentInfo
+login
+logout
+addPayment
+cancelPayment
Movie
-MovieID
-MovieInfo
-MovieName
-MoviePrice
+getTheater(MovieID)
+getShowtime(MovieID)
+getSeatInfo(MovieID)
+getTicketInfo(MovieID)
Reservation
-ReservationID
-ReservationInfo
+getSeats(MovieID)
+getAvailableSeats
(MovieID)
+getUnavailableSeats
(MovieID, SeatID)
Staff
-StaffID
-Name
-PhoneNumber
-Password
+login
+logout
Theater
-TheaterID
-TheaterLocation
-TheaterAvailableTime
-TheaterStartTime
-TheaterEndTime
Showtime
+MovieID
+PlayDate
+StartTime
+SndTime
Seat
-SeatID
-SeatInfo
Ticket
-TicketID
-TicketType
-TicketPrice

(d)


Customer
C: CustomerID
M: MovieInfo
B: Booking System
P: Payment System
D: Database
login(CID)
authorize(CID, password)
authorization
login success
viewMovieInformation()
getMovieInfoFromDB()
Movie info(description, theater, and show time)
Show movie info
bookingTickets(MID)
getAvailableTimeAndSeats(MID)
Current available time/seats
Show current available time/seats
selectPerferTimeAndSeats(MID)
saveBookingInfo(MID, CID)
Booking Info
Show booking info and go to payment page
payForTickets(BID)
verifyPaymentInfo(BID)
savePaymentInfo(BID)
Booking info with payment successful info
Show booking info with payment successful info