瀑布模式 System Develop Life Cycle

2022年9月4日 星期日　　上午4:00

瀑布模式定義：
瀑布模式又稱系統開發生命週期(System Develop Life Cycle)，該方法把系統開發的過程分為幾個階段，每個階段清楚定義要做哪些工作及交付哪些文件，各階段循序執行且僅循環一次，若在各階段發現錯誤時，允許階段間回饋，如此能儘早修正以減少系統修改或重做成本。

三階段瀑布模式
瀑布模式在階段劃分上較有彈性,當問題較小或較單純時可劃分為:分析、設計、導入

★ 五階段瀑布模式

(1)規劃(Planning)
　　通常由對IT部門的需求開始,稱為系統需求(System Requirement)，描述資訊系統或企業流程的問題，以及想要的調整。
(2)分析(Analysis)
　　建立系統的邏輯模型(Logical Model),第一步進行需求塑模(Requirement Modeling),最終交付成果為系統需求文件。
(3)設計(Design)
　　創造出滿足所有需求的實體模型(Physical Model),設計使用者介面，以及必要的輸入、輸出、流程與內外部控制。
(4)建置(Implementation)
　　目標為交付一功能完整、運作正常且具完整文件的系統,包含系統評估(System Evaluation),以確保系統是否運作正常、成本與效益是否符合預期。
(5)維運(Operation and Maintenance)
　　系統上線後,後續的維護、加強與保護。

★ 瀑布模式優點

(1)各階段皆考量完整需求，前一階段完全成功才會進入下一階段，較為嚴謹。
(2)每階段的產出都有確認、驗證、測試，以確保品質。
(3)正式申請才可以變更需求，具有正式版本控制。
(4)開發人員有明確的責任歸屬
(5)開發過程結構化且容易管理

★ 瀑布模式缺點

(1)專案開始時，User可能無法完全清楚描述需求。
(2)無法於各階段同時考量所有需求。
(3)一旦需求變更，文件要大幅修改。
(4)開發週期較長且過程中使用者參與不足
(6)程式撰寫於開發週期後段才開始，失敗風險高。

瀑布模式一般適用於低風險的專案，例如發展期間需求可清楚完整表達、需求較少改變或不會改變，問題領域知識容易取得、解決問題之資訊科技與設計方法成熟等。
一般來說編譯器、作業系統、程式語言等開發工具等軟體開發專案較為適用；大部分需求需介於終端互動產生之系統，因使用者可能無法清楚或完整提出需求，則可能無法有效執行

漸增模式 Increment Model

2022年9月4日 星期日　　上午4:13

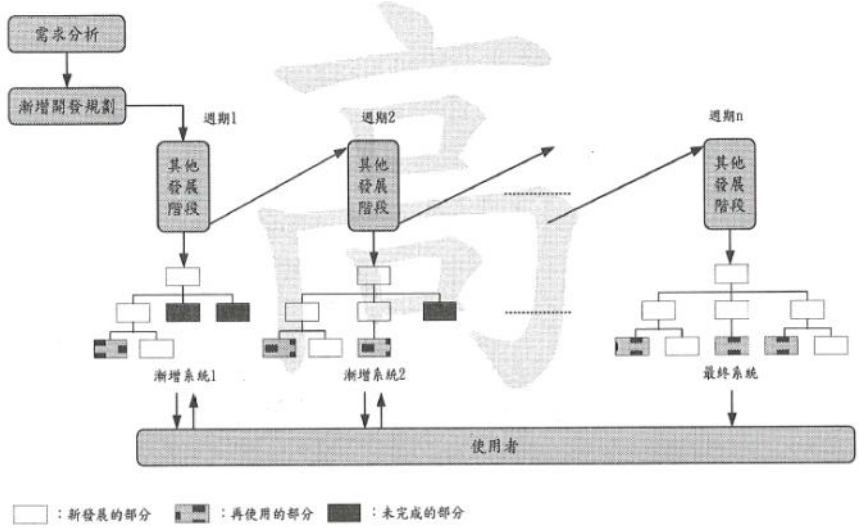
漸增模式(IncrementModel)定義：
把需求切割為子系統或子功能，之後依漸增開發計畫將切割的部分需求之開發訂為一個開發週期，每個開發週期依循瀑布模式於各階段清楚定義工作及交付文件，循序進行各階段且僅循環一次。漸增發展指每一次都在系統上增加新功能、模組、元件或子系統等。

漸增模式適用於:
(1)組織目標與需求可完全且清楚地描述
(2)可先做系統整體規劃,預算再分期編列並執行
(3)應用漸增模式讓組織有充裕的時間來學習與轉移技術

漸增模式vs.瀑布模式比較
1.子系統:
　瀑布模式為同時開發；漸增模式為獨立依序開發
2.開發周期:
　瀑布模式僅有一個開發週期，所有項目必須同時開發；漸增模式為多個開發週期，可分次進行
3.開發階段
　瀑布模式與漸增模式皆是完整的開發階段，循序進行至循環一次結束
4.系統風險
　因瀑布模式程式編輯是在後期階段開始，故風險較高，此風險包括資訊科技的改變、需求變更、關鍵人員流失，因此軟體失敗的可能性較高；相較漸增模式每個週期皆有程式編輯及上線實施，使用者皆有參與，故漸增模式的風險較低

漸增模式 vs. 瀑布模式

	瀑布模式	漸增模式
子系統	同時開發	獨立依序開發
開發週期	一個	多個
開發階段	完整	完整
系統風險	較高	較低



雛型模式 Prototype Model

2022年9月4日　　下午 01:10

雛型模式(Prototype Model)定義：

由於使用者常無法將需求清楚且完整地表達，或雖可清楚地表達，但資訊人員可能沒有夠的經驗與知識完整地瞭解使用者需求，或無法一時之間找出解決問題的方法、模式或資訊科技等。雛形模式先針對使用者需求較清楚的部分或資訊人員較能掌握之部分，依分析、設計與實施等步驟快速開發雛型。

雛型模式

(1)強調雛型之快速開發及使用者高度參與。
(2)以雛型作為使用者及系統開發者之需求溝通與學習工具。
(3)從需求最清楚的部分著手開發雛形，並透過使用者意見回饋反覆修改與擴充，盡可能縮短每次反覆時間間隔。

雛型模式的優點:

使用者於整個系統開發過程中高度參與,有助於使用者之需求創新與表達、資訊人員對需求之瞭解與掌握、增進雙方溝通、使用者對系統之接受度與滿意度、新問題之瞭解、新方法之衍生、新資訊科技之學習與應用等。

雛型模式的缺點:

(1)強調以雛型演進代替完整之系統分析與設計，故文件較不完備，程式亦可能較難維護，短期而言可能較能滿足使用者需求;但就長期而言系統較易失敗。
(2)缺乏整體之規劃、分析與設計故較不適用於大型及多人參與之系統開發專案。

雛型模式的適用範圍:

雛形模式適用於需求可能改變於整個專案生命期間、客戶能高度參與、應用領域不熟悉或高風險等情況的專案，較不適用於具有固定需求與實際技術上沒有困難的專案。

演進式雛型(EvolutionaryPrototyping)

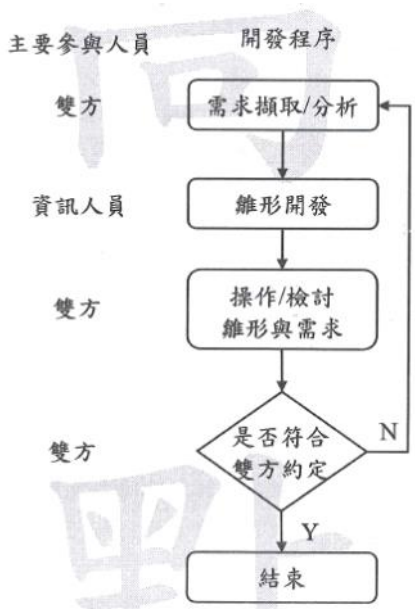
將所有需求看成一個整體,從需求最清楚的部分先快速經歷一系統開發週期,以完成初版雛型系統之開發。再利用該雛型與使用者溝通,以確定、修改和擴充需求並藉以作為下一週期雛型演進之依據。不斷地反覆進行,直到雛型系統符合雙方約定為止

適用於對雛形模式有經驗的開發者,因此開發方法需使用者與資訊人原有良好的溝通與專案管理,否容易生系統功能不符合使用者需求的問題。

用後丟棄式雛型(RapidThrowawayPrototyping)

以快而粗糙的方式建立雛型,讓使用夠盡快藉由與雛型之互動來決定需求項目,或允許資訊人員藉以研發問題之解決方法與資訊科技之應用等。因用後即丟,所以不需考慮雛型系統的運用效率、可維護性與容錯能力。

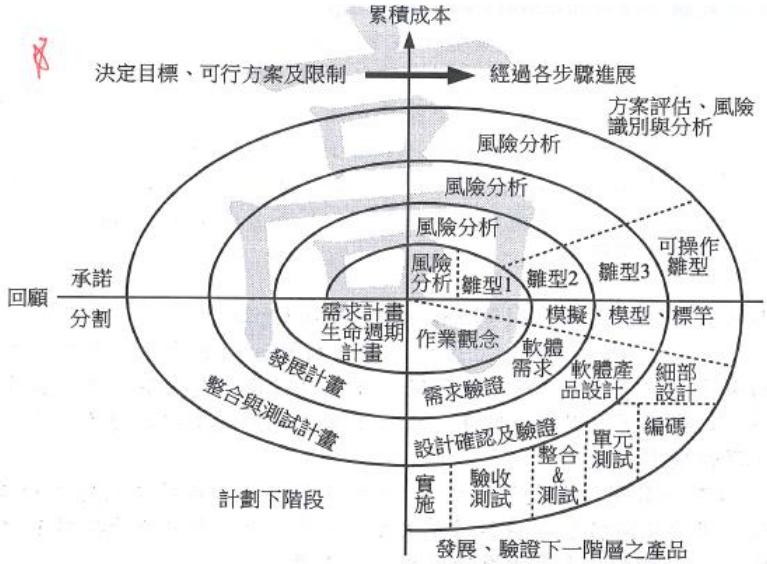
僅實施於風險程度最高之處,如在使用者需求或解決問題之知識、概念與資訊科技整合最不清楚之處”除此之外,盡可能地採用演進式雛型,因雛型之丟棄也意謂著成本的浪費



螺旋模式 Spiral Model

2022年9月4日　　下午 01:26

螺旋模式(Spiral Model)定義：
由於漸增模式與雛形模式皆無法完全解決瀑布模式執行上的問題，故基於瀑布模式應用於政府大型軟體專案之經驗，經多次修改而成螺旋模式。此模式之執行由三個步驟形成一週期，此週期反覆進行直到系統開發完成為止，其中每週期的進行均強調規劃及風險評估。



資料來源："系統分析與設計(六版)"，吳仁和、林信惠

步驟一:找出系統目標、可行方案與限制
(l)找出系統目標: 如系統績效、功能與容忍改變之能力等。
(2)找出可行方案: 因地制宜找出系統的可行方案。
(3)可行方案限制:如專案之成本、時程、系統介面等

步驟二:依目標與限制評估方案
(1)找出不確定的部分亦即專案風險之重要來源。
(2)解決風險來源。
可用雛型、模擬、標竿(Benchmarking)、參考點檢查(ReferenceChecking)、問卷、分析模式等技術以解決風險" 選擇風險解決方法時應考慮成本效益。

步驟三:由剩下之相關風險決定下一步
(1)若績效或使用者介面風險將強力影響程式開發或內部介面控制，則下一步可採演進式雛型。
(2)若該雛型使用性佳且夠強韌(Robust)，足以當作未來系統發展之基礎，則往後將可採一系列的雛型演進。
(3)假如先前之雛型已解決所有的績效或使用者介面之風險，且程式開發及介面控制之風險獲得掌控，則下一步將遵循基本的瀑布模式，亦可適當地修飾以整合漸增模式

螺旋模式特色
(1)每個週期的結束需由系統相關人士或組織來檢討系統績效,內容包括上一週期所有的系統發展、對下一週期的規劃及所需資源等。
(2)檢討目標是確保所有相關組織皆會相互承諾下一階段之方式
(3)下一階段之規劃，可能包括將系統分割為幾個後續發展的部分或是由其他組織或個人之開發元件

螺旋模式應用原則
(1)在高風險部分之設計尚未穩定前，規格之發展不需要一致、詳盡或正式，以避免不必要之設計修改
(2)於開發任一階段，螺旋模式可選擇整合雛型模式以降低風險。
(3)當找到更吸引人方案或需解決新風險時，螺旋模式可整合重做或回到前面之階段。

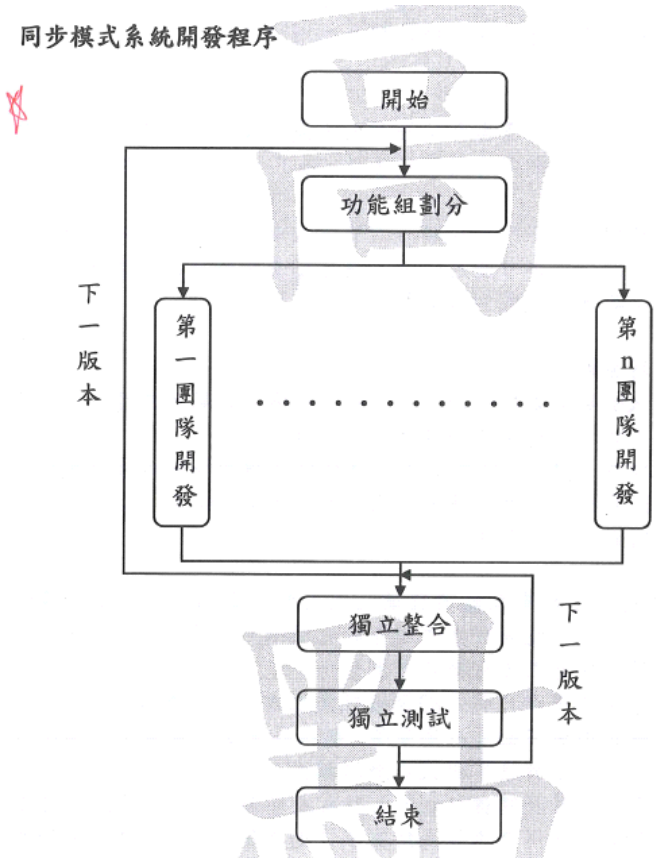
螺旋模式包容了現有軟體流程模式大部分優點,且其風險導向之方法解決了許多系開發模式所存在之問題。
在某些條件下,螺旋模式相當於某一現有之流程模式。

例如:
(1)若專案在使用者介面綜合績效求方屬於低風險，且在預算及時程控制方面屬於高風險，則這些風險之考量會使螺旋模式之執行相當於瀑布模式或漸增式。
(2)若專案在預算及時程控制、大型系統整合之風險較低，且在使用者介面或使用決策支援需求方面之險較高，這些風險之考量會使螺旋模式執行類似於雛型模式

同步模式 Concurrent Model

2022年9月4日　　下午 10:06

同步模式(ConcurrentModel)定義:
將每一版本(Release)的工作分為數個功能組(Eancement)，其為一或多個功能的組合。接下來將功能組的工作分派給數個開發團隊平行開發。當同一版本的功能組都完成後，便交由獨立的團隊進行整合與測試,開發團隊可進行下一版本的開發，當整合與測試團隊完成一個版本的工作後，便可進行下版本的整合與測試。



同步模式構想:

(1)活動同步(Activity Concurrency)
將系統專案工作切割讓工作團隊平行進行，以增加工作效率。
(2)資訊同步(Information Concurrency)
讓不同團隊間的資訊得以相互交流與共享，可分為：
1.向前傳遞(FrontLoading)
將下一階段的重要議題及相關考慮因素，提前讓前一階段的開發團隊知道。
2.向後傳遞(Flying)
將前一階段開發狀況及重要資訊傳給下階段開發團隊。
3,建立一個有效的資訊交換網路及支援群體工作的環境。
(3)整合性的管理系統
同步模式的管理必須開發一個管理系統以協調人員、資源、過程及產品間複雜的互動關係。

★ 同步模式的優點

(1)縮短系統開發時間，加速版本更新
(2)提高產品競爭力以因應市場

★ 同步模式的缺點

(1)緊湊的步驟及資訊溝通的頻繁
(2)專案管理的成本與複雜度大增

★ 同步模式的目的

同步模式的目的是為了縮短開發時程，只要有足夠的人力及良好的管理制度，即可應用於任何一種資訊系統。
由於商業套裝軟體的競爭激烈且版本更新的壓力大，故同步模式較適合商業套裝體的開發，較不適用於客製化軟體。

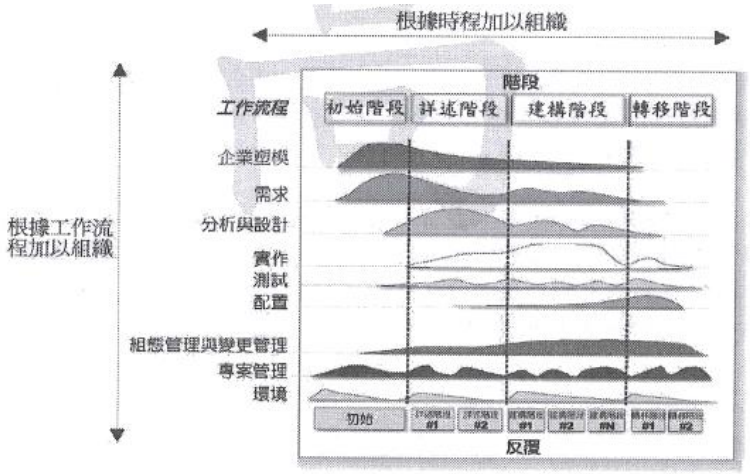
RUP模式(Rational Unified Process , RUP)

2022年9月4日　下午 10:18

RUP模式(Rational Unified Process , RUP)定義:

結合螺旋模式的概念，以反覆與漸增的原理進行軟體開發，且每一次的反覆後需產出一個可運作的系統版本，並在每一個反覆週期中評估風險，以盡早發現問題。反覆指重複用相同的一系列操作或步驟進行軟體開發。

RUP可由動態與靜態兩個構面來說明系統開發專案之實施階段與核心工作，水平軸與垂直軸交叉格上的圖形面積代表其所對應工作之估計工作量或比率，然而並非完全固定，可能因專案性質、週期、團隊經驗及其他經驗而有所調整。



資料來源："系統分析與設計(六版)"，吳仁和、林信惠

★ 動態面

為水平軸，將軟體開發依序分成下述週期階段，週期與其階段可視情況反覆進行。

(1)初始階段(Inception)

目標為建立系統需求與範圍、接受準並評估整體風險、構想企業個案，取得參與人員認同。

由於需為系統建立商業案例並確定專案範疇，因此必須識別所有與系統進行交互作用的外部實體,並定義交互作用的特性所關注的是整個專案進行中的業務和需求方面的主要風險。

里程碑為完成生命週期目標(Lifecycle Objective)，其用來評估專案基本的生存能力。

(2)詳述階段(Elaboration)

目標為處理主要的技術工作與探討技術風險，分析問題領域建立健全的體系結構基礎，編製專案計劃，淘汰專案中風險最高的元素。

因此必須在理解整個系統的基礎上,決定體系結構的範圍、主要功能和諸如性能等非功能需求。同時為專案建支援環境,包括創建開發案例,創建模板、準則並準備工具。里程碑為完成生命週期結構(LifecycleArchitecture)，其為系統結構建立了管理基準，並使專案小組能夠在下一階段中進行衡量，必須檢驗詳細的系統目標和範圍、結構的選擇以及主要風險的解決方案。

(3)建構階段(Construction)

目標為建構一初步可運作的系統版本(α版)。將所有仍留存的物件和應用程序功能開發並組成產品，並完成所有的功能被詳細測試。其重點放在管理資源及控制運作以最佳化成本、進度和品質

里程碑為由α版演化成具有完整功能的系統版本β版)，決定了產品是否可以在測試環境中進行部署。

(4)轉移階段(Transition)

目標為依使用者回饋精調系統、組態、安裝與可用性,建立最 終版本的系統並移交給客戶使用。

為確保系統品質，可跨越數次的測試與修改。里程碑為交付使用者軟硬體系統、使用者手冊、問題排解指南，將系統正式上線。

(5)生產階段(Production Phase)

對正在使用中的軟體進行監測、提供操作環境(基礎設施)的支援，和缺點的報告與提出或評估改變的要求。

★ 靜態面

為垂直軸,主要處理依邏輯順序將軟體開發與管理支援工作表達成下述九個核心工作流程。其中，前六項為軟體工程工作，而後三項則為管理支援工作。

(1)企業塑模(Business Modeling)

瞭解系統要部署的目標組織之未來結構與動態，瞭解其目前問題與找出可能的改善方式 ，確保客戶、終端使用者與開發者對目標組織有共同的瞭解，並產出企業模型。

(2)需求(Requirement)

建立與維護客戶及其他參與者對系統應做什麼(What)與為何做(Why)之認同，如定義系統範圍、產出系統需求、規劃技術內容、評估系統開發成本與時程等。

(3)分析與設計(Analysis and Design)
將需求轉換成如何實作系統之規格，需描述對需求之瞭解，及如何選擇最佳的實施策略將需求轉換成系統設計。

(4)導入(Implementation)
將子系統之組織階層定義程式碼之組織，以元件實作類別與物件，並對各元件進行單元測試，將成果整合成可執行的系統。

(5)測試(Test)
發現與紀錄軟體問題，藉由需求與設計規格，及具體的系統評估，驗證軟體是否能設計運作、需求是否被適當的實作等。

(6)部署(Deployment)
測試軟體在最終作業環境之運作(β測試),包裝軟體以便交付、配送與安裝軟體、訓練終端使用者與銷售人員、移轉(Migrating)現有軟體或轉換(Coverting)資料庫等。

(7)組態與變更管理(Configuration and Change Management)追蹤與維護專案資產在瀕進過程之完整(Integrity)。在軟體發展生命 週期中,會有許多有價值的產出，亦重要資產,需被安全地看管以隨時再利用。

(8)專案管理(Project Management)
提供管理軟體專案的架構,提供實務準則以供規劃、人員訓練、執行與監督專案，提供管理風險的架構。然RUP模式並不包含所有專案管理之議題，如人員、預算、合約管理等。

(9)環境(Environment)
以流程與工具支援軟體開發之組織，包含選擇、取得與配置工具、處理組態與改善技術服務以支援流程等。

RUP模式塑模元件
(1)工作人員(Worker)
參與專案的人們在專案中扮演的角色(Role)，例系統分析師、專案經理等。其定義每參與者在專嘹-中所做之流程工作、應具備的才能與應負的責任
(2)活動(Activity)
是一位特定工作人員在角色上所執行一個工作單元，活動有清楚的目標，通常會產生或更新工作產出，例如模式、類別或計畫。
(3)工作流程(Workflow)
一連串活動的組合，會產生有價值或有意義的結果。工作流程通常會描述活動之順序，顯示工作人員所參與之活動及彼此間之互動。
(4)產出(Artifact)
經由活動或工作流程所製造、修改或使用的資訊。產出為專案的實際產品,可能是輸入資訊、輸出資訊，或是運算結果。

RUP三大特色
(1)使用案例驅動(Use Case Dnven)
以使用案例為核心，對軟體開發各階段所參與的角色，定義其應有之責任與活動，從使用者的角度捕捉其需求，並轉換成系統功能。開發程序有良好的流程支援，且受到好的工具支援。此外，可將整個專案，依用功能需求切割成小的子專案進行開發。

(2)架構中心 (Architecture-centric)
RUP大部分聚焦於塑模(Modeling)，模型可作為使用者與開發者間的溝通橋樑，並幫助開發者瞭解與形塑問題與解決方案。此外，
模型為真實世界的簡化表示，可幫助瞭解不容易透徹理解的複雜系統。好的模型近似於真實世界，並相互協調以確保其一致(Consistent)
RUP目的為找出可達成使用案例功能的模型架構，且為可瞭解(Understandable)可維護(Maintainable)與可再用(Reusable)。

(3)反覆與漸增(lterative and lncremental)
是一種持續探索(Discovery)、創造(Invention)與導入(Impmentation)的方法，每次的反覆促使開發團隊驅動產品更為接近可預測與重複的方向。一反覆為完整的開發循環，產出可執行產品的釋出版本，其為開發過程中最終產品的子集合，從每次的循環中逐漸成長，以致最終系統。

★ RUP的優點

(1)結合許多成功的方法論，軟體開發流程完整
(2)可使用UML作為共可的溝通語言
(3)反覆式的流程可降低開發風險

★ RUP的缺點

(1)因包涵廣泛,學習較為困難。
(2)開發的時間成本較大

敏捷軟體開發(Agile Software Development)

2022年9月5日　上午 12:16

敏捷軟體開發(Agile Software Development)定義:

★ 敏捷宣言(Agile Manifesto)

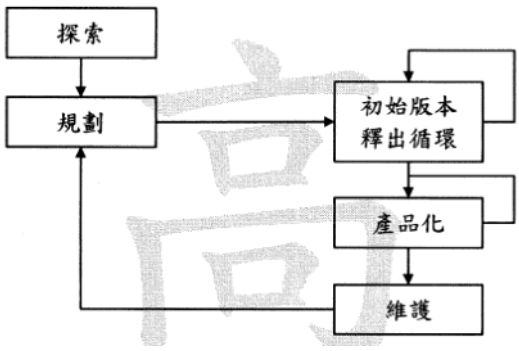
由一群不同軟體開發方法的領域代表所共同發表，主要目的為提出一套較傳統軟體開發方式更為簡捷且快速的軟體開發概念。
其主要開發理念和價值觀如下(Becketal. Et al.,2001):

(1)因應變化勝於遵循計畫。
(2)個體與互動勝於流程與工具。
(3)可運作的軟體勝於全面性的文件。
(4)與客戶的協同合作勝於契約談判。

★ **敏捷開發原則**

(1)透過及早並持續地交付有價值軟體讓客戶滿意。
Customer satisfaction by early and continuous delivery of valuable software
(2)歡迎改變需求，甚至處於開發後期亦然。
Welcome changing requirements, even in late development.
(3)頻繁交付可用的軟體，週比月更佳。
Delivrer working software frequently (weeks rather than months)
(4)業務人員與開發者必須在每天密切合作。
Close,daily cooperation between business people and developers
(5)以積極、可信任的人來建構專案。
Projects are built around motivated individuals, who should be trusted.
(6)面對面的溝通是最棒的溝通方式(同地點)
Face-to-face conversation is the best form of communication(co-location).
(7)可運作的軟體是主要的進度衡量方法。
Workingr software is the pritnary measure of progress.
(8)持續地開發，維持穩定的步調。
Sustainable development,able to maintain a constant pace.
(9)持續追求優越的技術與良好的設計。
Continuous at tention to technical excellence and good design
(10)精簡-最大化未完成工作量之藝術-是不可或缺的·
Simplicity-the art of maxirmzing the amount of work not done-is essential.
(11)最佳的架構、需求與設計皆來自於能自我組織的團隊
Best architectures, requirements, and designs emerge from self-orgamzing teams.
(12)團隊定期自省如何更有效率，並調整之。
Regularly, the team reflects on how to become more effective,and adjusts accordingly.

敏捷開發核心流程



★ **敏捷開發核心流程:**

(1)探索(Exploration)
探索環境、評估問題以及解決方案、組成團隊,以及評估團隊成員技能，時間可能從數週(對成員與科技皆熟悉的狀況)至數個月(一切事物皆為新之基礎)。
(2)規劃(Planning)
你與客戶需同意，在接下來的專案時間內緊密地合作，直到最終解決方案上線為止。如果探索階段有效率的話，此階段可能花費數天的時間。
(3)最初版本釋出循環(Iterations to the First ReIease)
為測試(Testing)、回饋(Feedback)與變更(Change)的循環,通常為數週的時間
(4)產品化(Productionizing)
加速上階段的插還，可於每天報告進度與成果，而當增加新功能時，亦可快速釋出新版本。
(5)維護(Maintenance)
系統上線後,後續仍須持續釋新版本，保持其平順地運作。於此階段，團隊可能輪替、或是團隊成員有所改變。

敏捷開發控制變數:

(1)時間(Time)
必須有足夠的時間來完成專案，包含傾聽客戶、設計、編碼，以及測試的時間。
(2)成本(Cost)

評估傾聽客戶、設計、編碼,以及測試的成本，其牽涉到與時間、範疇以及品質的平衡

(3)範圍(Scope)

範疇為傾聽客戶並寫下其故事後所得之產物，用來調查滿足客戶需求之功能，故事應簡短且易於掌握重點。

(4)品質(Quality)

理想的系統品質為完美，且後續易於維護。然而需權衡品質與上述三者之連動性

★ 敏捷開發核心活動

(1)客戶現場(Onsite Customer)

客戶與專案團隊一起工作，專案團隊到客戶現場進行開發，或是邀請客戶到專案團隊公司裡來進行開發。如果開發過程中出現問題或是新產品發佈後，能夠以最快速度得到客戶的回饋並進行改善。

(2)一週工作40小時(40-hour Work Week)

強調在上班時間內，團隊緊密合作認真工作，並在下班後放鬆、疏壓，以減少過勞或工作效率不佳而造成的錯誤或缺失。

(3)頻繁發佈(Short Releases)

強調儘量頻繁的產品發佈，一般以周、月為單位。如此一來，客戶每隔一段時間就會取得發佈的新版本並提供反饋來改進產品。由於發佈頻繁，每一個版本新增的功能簡單，簡化了系統文件和設計。又因設計簡單，所以當客戶有新的需求或者需求變動時，也能快速適應。

(4)配對編程(Pair-Programming)

在敏捷開發中，做任何事情都是配對的，包括分析、撰寫測試編碼、撰寫實際編碼或重構。其優點為兩個人在一起討論很容易產生思想的火花，也不易偏離正軌。

敏捷開發其他活動

(1)測試驅動開發(Test-DrivenDeve10pment)

為敏捷開發最重要的部分,任何一個系統功能都是由測試開始。首先對需求進行分析,分解為一個個的故事(St。,並記錄於故事卡(St。Card)。之後兩個人同時坐在電腦前,一人依照故事來撰寫測試編碼,另人如有不同的意見則進行討論,直到達成共識為止。接著由另人撰寫實際的功能編碼。

(2)持續整合(ContiriUOUShtegration)

傳統的軟體開發,通常需置過長時間才會做整合,如此一來會引發很多問題,例如整合失效。敏捷開發提倡持續整合,一天內整合十幾次甚至幾十,如此頻繁的整合能盡量減少衝突”此外,由於整合頻繁一莓整人的變動也很少,即使整合失敗也容易找到錯誤e

(3)重構(Refactoringy

在不改變系統外行為下,對系統內=部結構進行最佳化,使編碼儘量簡單、優美、可擴展"在敏開發中,重構貫穿於整個開發流程,每次開發者簽入(Checkin)編碼前,都要對所寫編碼進行重構。值得注意的是,每次重構時改變要盡可能小,並用單元測試來驗證其是否引起衝突。

(5)站立會議(StandUpMeetmg)

專案的所有成員每天早上進行一次站立會議,時間約15-20分鐘·會議的內容非為需求分析、任務分配等,而是每個人都回答三個問題:你昨天做了什麼?你今天要做什麼?你遇到了哪些困難?站立會議讓團隊進行交流,彼此相互熟悉工作內容,如果有人曾經遭遇類似問題,可在會議後進行交流·

(6)最小化文件(Mit1cumentati)

敏捷開發中主要的文件為試編碼,其真實反應了客戶需求與系統AP I 用法;而言好的敏捷開發碼不需大量的註解(Comment),簡單易讀的碼才是好的編碼。如果編碼架構過於複雜,則需進行重構

(7)聚焦合作(CollaborâiiVef6cus

敏捷開發中,每個人都有權利獲得系統任一部分的編碼並修改之,而不需徵編碼作者的可、此一來,每個人都能熟悉系統的編碼,即使專案團隊的人員變動也沒有風險”

(8)自動化測試(AutomatedTesting)

為了減小人力或者重覆勞動,所有的測試包括單元測試、功能測試與整合測試等都是自動化,QA人員需熟悉開發語言、自動化測試工具,並能撰寫自動化測試腳本或者用工具錄製。

(9)適應性計畫(AdaptivePlanning)

敏捷開發的計畫是可調整的,而非像傳統開發過程每階段都是有計畫的進行,一階段結束後再開始下階段。敏捷開發只有一次次頻繁的新功能發佈,並依據客戶回饋隨時作出相對應的調整和變化·

敏捷開發特色

(1)適應性(Adaptive)而非領設性(Predicti、”

軟體開發的困難在於體需求的不穩定,導致軟體開發過程的不可預測。傳統開發方為針對一個軟體開發專案,於長時間做出詳細的計畫,然後依計畫進行開,因此在很難適應環境的變化,甚至拒絕變化捷開發方'則是歡迎變化、適應變化,稱為適應性方法

(2)人導向(PeopleOnented)ffJ#流導向orocessOnented)。

MatinFwer認為在敏捷開發過程中首重人而過程其次。於傳統的軟體開發分配工作的重點明確定義角色,以個人的能力去適應角色,角色是為了保證流程的實施,人可取代而角色不可取代。要求研發人員獨立自主在技術上進行決策,並特別重視專案團隊中的訊息交流。

敏捷開發誤解

(1)對人的要求很高

很多人在嘗試實施敏捷開發時會說:對人的要求太高了,我們沒有這樣的條件也沒有這樣的人,因此沒辦法進行敏捷開發。

敏捷開發對人的要求並不高,並可以培養開發人員所需的各種能力,前提是其處在真正敏捷的環境中。

(2)沒有文件也沒有設

鼓勵“非到必要且意義大時不文件然判斷標準取決於客戶,實務上可由專案團隊協調,盡量避免個人獨斷。此外,應持續設計,將設計工作分攤到每天的日黨工作中,不斷的設計、重構,使設計一直保持靈活可靠。

(3)只有敏捷開發好

軟體開發方法沒有好壞,取決於是否適合解決問題。每一種方法都有其最善於解決的問題和最佳的發揮環境,需求穩定適用瀑布模式;而敏捷開發適用於變化快風險高的軟體開發專案。

(4)敏捷開發就是極限編程

極限編程只是眾多敏捷開發方法之一,還有很多其他的敏捷開發方法,其背後的理念和原則都是敏捷宣言。

(5)所有的專案都要遵循制定的敏捷開發標準

應該為專案團隊找到最適合的開發方法,而不是先確定開發方法再讓專案團隊適應。即使同一個開發團隊,隨著專案的進行,對需求、技術理解的慢慢深入,一開始適合專案的方法也可能漸漸不合適,此時就要進行即時的調整。敏捷開發是動態的,而非靜止不變的,因為世界本身就不停的在改變。

動態開發方法(Dynamic System Development Method,DSDM)

2022年9月5日　　上午 12:49

動態開發方法(DynamicSystemDevelopmentMethod,DSDM)定義：

起源於90年代中期,為RAD開發方法之延伸,強調使用者在開
發過程中的參與互動,開發過程主要以反覆與漸增的方式進行。主要
工作分述如下:
(1)專案前(PreProject)
提出專案建議並選擇之
(2)專案生命週期(ProjectLife-Cycle)
I.可行性研究(FeasibilityStudy)
2.企業研究(BusinessStudy)
3.反覆功能塑模(unctionalModelIteration,EMI)
反覆設計與建置(DesignandBuiIdIteration,DBI)
5.導入(Implementation)
(3)專案後(Post-Project)
衡量系統績效並提出需加強之處。

Scrum軟體開發

2022年9月5日　　上午 12:50

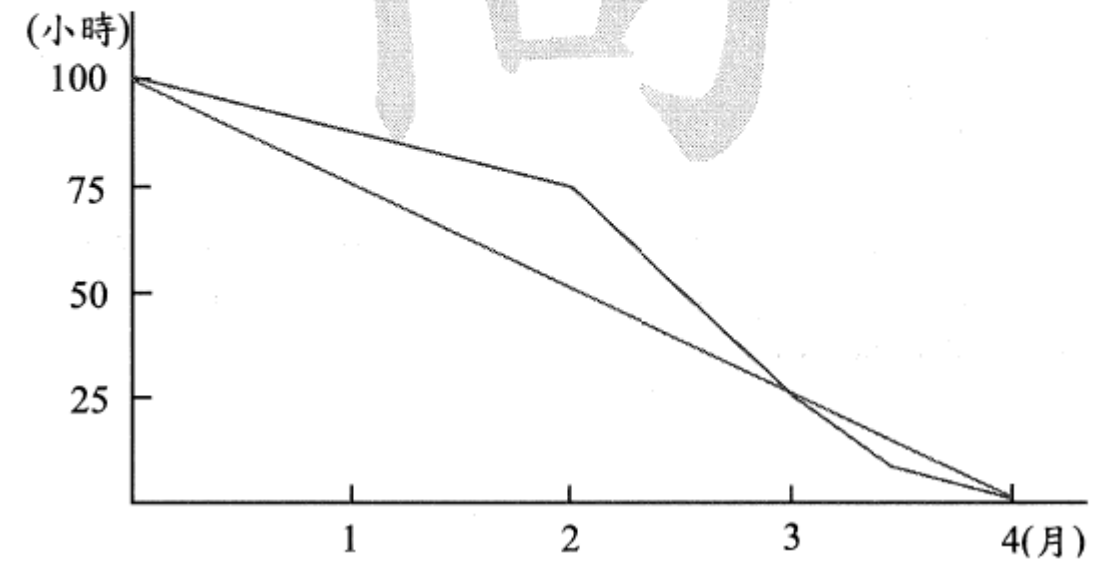
Scrum軟體開發定義：
強調反覆地短期發展、檢討與調整,以自我組織與管理目標來建立開發團隊,主張以固定時間長度的衝刺(Sprint)進行系統開發,並在每一衝刺結束時展示所完成的功能。

衝刺準備階段
(1)組成Scrum團隊
1.產品負責人(ProductOwner)
包含負責訪談客戶定義產品需求並承擔成敗責任。
2・Scrum專家(ScrumMaster)
領導開發團隊依循Scrum精神來發展品,並確保scrum流程內的活動依正確的步驟進行
3,開發團隊
由6~10人組成擁度自主權的跨功能團隊”
(2)需求擷取
客戶主動提出,是從訪談內容擷取出需求"
(3)需求轉換
將使用者需求轉換成故事,這些故事的集合稱為產品清單(ProductBacklog),並依其重要性排序。
故事是Sc八Ⅶ的一個基本分析單位,其以使用者觀點定義出一些特色、一條執行路徑或是一個情節(Scenari。),且每個故事的長度大小須可在一次衝刺中完成。
故事的表達方式如下:
身為唄色丿,我可以做麼行為丿,以達到什麼目的

衝刺進行階段
(1)衝刺規劃
產品負責人召開衝刺規劃會議,從產品清單中挑選此次所要開發的需求,並細分為工作項目與估算時程。會議結束後,scrum專家撰寫衝刺資訊文件,內容含要製作的故事,衝刺目標、起迄時間,以及參與的團隊成員(陳建村012)·
(2)衝刺執行
進行為期1~4週的開發活動·含設計、編碼、整合與測試等。每天需召開約15分鐘的站立會議,並搭配燃盡圖來管控工作時數”
(3)衝刺檢O(SprintRevrew)
整個衝刺週期結束後,召開衝刺檢視會議,將成果展示給產品負責人(Keith,20川)。
(4)衝刺評估
產品負責人召開衝刺自省會議(SptRetrospectiveMeeting),檢討與改進專案的軟體開發流程,並提出改善方案(Keith,2010)。會議後由Scrum專家撰寫衝刺總結報告,最後召開產品清單精進會議,總結問題與經驗。
(5)重複衝刺
直到完成產品清單所有故事

燃盡圖(Burn Down Chart)

表示剩餘工作量的工作圖表，橫軸代表時間，縱軸代表剩餘工作量，可直觀地預測工作將於何時完成。以學生成績系統之計算成績模組為例，其需 4 個月的開發時間，並投入 100 小時的開發人力，如下：



精實軟體開發

2022年9月5日　　上午 12:54

精實軟體開發

將豐田生產系統(ToyotaProductiveSystem,TPS)提出之精實生產(LeanManufacturing)方法,應用於軟體開發領域。包含:
(1)消除浪費(Eliminatewaste)
找出軟體開發過程中浪費的步驟並消除之。
(2)增進學習(AmplifyLearmng)
可用短期且完整循環的反覆開發來取得回饋並增進學習。
(3)延遲決R(DeIayCommitment)
依據事實做最終決策避免不確定的結論。
(4)快速i&i&(DeliverFast)
使用者一旦確需求後,以平穩、快速的流程來回
應客戶需求。
(5)團隊授權(Empowertheteam)
開發團隊應自行設立目,決開發流程、蒐集資料等來自我
要求與決定,以成目標。
(6)建置完整(BuildIntegrityln)
含開發者徹底瞭解使用者或企業需求的感知完整(PerceIVed Integrity);以及系統的獨立元件皆能完整運作,提供使用者一功能完整之系統的概念完整(ConceptualIntegnty)。
(7)全盤檢視(Seethe。le)
開發者需全檢視是否達成使用者的整體需求。

極限編程(Extreme Programming)

2022年9月5日　　上午 12:55

極限編程(Extreme Programming)定義:
著重於開發流程是否對使用者或企業產生價值,強調以有效率且富彈性(反覆與漸增)方亡開發高品質之軟體系統。極限編程提出四項軟體開發基本行

(1)編碼(Coding)
系統開發過程最重要的產出為可運作的程式碼,其有助於找出適當的解決方案一一一

(2)測試(Testing)
藉由小部分的測試減少某些多餘的流程;藉由許多的測試過程減少更多累的流程一

(3)傾聽(Listemng)
開發人員必須傾聽使用者需求,並以技術觀點回饋。

(4)設計(Design)
若未經設計可能無法釐清系統範疇,易導致系統內協同運作的元件過度相依,即修改部分元件會影響其他元件的運作·