極限編程 XP eXtreme Programming

什麼是極限編程？(1/2)

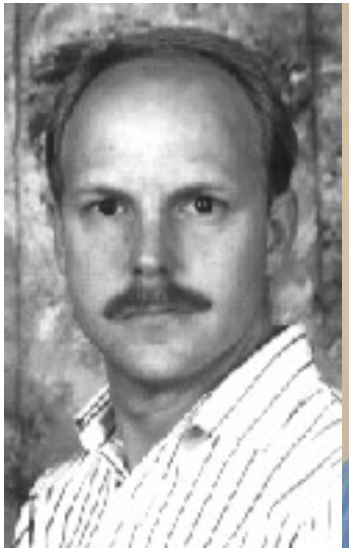
- 一種軟體工程方法學
- 敏捷軟體開發中最富有成效的幾種方法學之一
  - 強調程式設計師團隊與業務專家之間的緊密協作、面對面的溝通（認為比書面的文檔更有效）、頻繁交付新的軟體版本、緊湊而自我組織型的團隊、能夠很好地適應需求變化的代碼編寫和團隊組織方法，也更注重做為軟體開發中人的作用
- 傳統 V.S XP
  - 與傳統在項目起始階段，定義好所有需求再費盡心思的控制變化的方法相比， XP 希望有能力在項目周期的任何階段去適應變化

什麼是極限編程？(2/2)

- XP 其實它沒什麼新鮮的概念，而且大部分都是相當古老的概念。有些專家認為 XP 是保守的
  - 它所有的技巧都已在過去數十年或數百年內被證明有效
- XP 的創新處在於：
  - 把這些做法一舉囊括
  - 確保這些做法都儘可能徹底的實施
  - 確保這些做法都儘可能彼此截長補短

歷史

- 極限編程的創始者是 Kent Beck 、 Ward Cunningham 和 Ron Jeffries， 他們在克萊斯勒時提出了極限編程方法。
- 1999 年 10 月發行《極限編程解析》（2005 第二版出版） By Kent Beck



極限編程的哲學思想

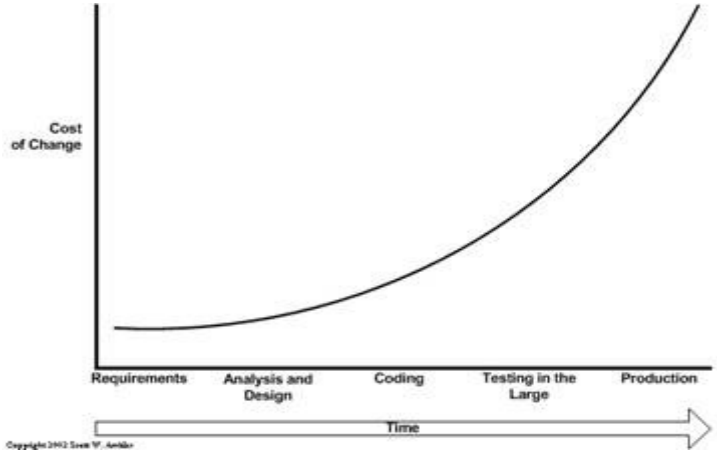
- 一種社會性的變化機制
- 一種開發模式
- 一種改進的方法
- 一種協調生產率和人性的嘗試
- 一種軟體開發方法

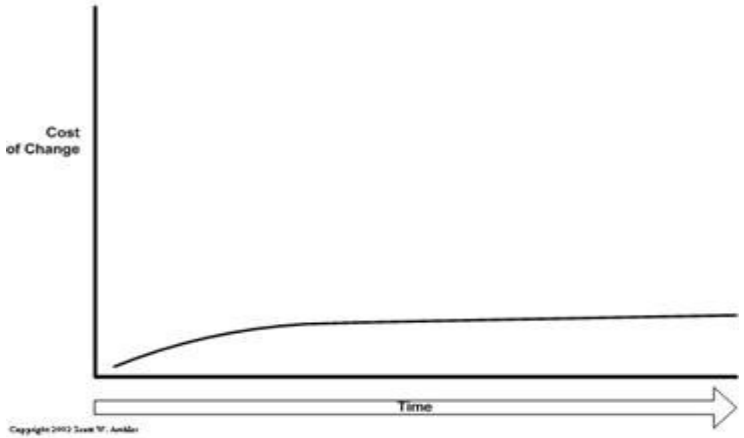
By 《極限編程解析》

XP 的目標

- 極限編程的主要目標在於降低因需求變更而帶來的成本
- 極限編程透過引入基本價值、原則、方法等概念來達到降低變更成本的目的

傳統　　V.S.　　XP





XP 的極致思維 (1/2)

- 如果程式碼檢查（code reviews）有益，我們應反覆地檢查（搭檔編程，pair programming）。
- 如果測試有益，每個人都該常常做測試（單元測試，unit testing），即使是客戶也不例外（功能測試，function testing）。
- 如果設計有益，則應被當作每個人每天工作的一部分（重整，refactoring）。

XP 的極致思維 (2/2)

- 如果簡潔有益，我們應讓系統保持在能夠支援目前所需功能的最簡單狀態（能夠運作的最簡單架構）。
- 架構如果重要，每個人都要常常反覆琢磨架構（對整個系統下一個隱喻、象徵、或概念）。
- 整合測試如果重要，我們每天就會做上好幾次（持續整合）。
- 短的開發週期如果有益，我們就把它縮短到真的真的非常短 --- 短到可以用幾秒到幾分鐘到幾小時來計算，而不是幾星期到幾個月到幾年（通盤規劃）。

XP 可保証二件事

- 對程式師而言
  - XP 保証他們可以每天都做些真正有意義的事；他們再不必獨自面對那些會令人驚慌的情況；他們將可以自己掌握每件事，成功地做出系統；下他們能做的最佳決定，而不會做不是他們夠資格做的決定。

- 對客戶和經理人而言
  - XP 保証他們每個工作週，都可以獲致最大的利益；每隔幾週，就會看到他們所在乎目標的具體進度；也可以在不導致過高費用的狀況下，在專案進行到一半時改變其進行方向。

By 《極限編程解析》

XP 的價值

- 溝通
- 簡單
- 回饋（設計師 & 客戶的測試）
- 勇氣
- 尊重（第二版最新添加的價值）

XP 的原則

- 由價值衍生而來
  - 快速反饋
  - 假設簡單
  - 增量變化
  - 包容變化

XP 的特徵

- 增量和反覆式的開發
  - 一次小的改進跟著一個小的改進
- 反覆性，通常是自動重複的單元測試
- 成對程式設計
- 在程式設計團隊中包含 User
- 軟體重構
- 共享的程式碼所有權
- 簡單
- 回饋

XP 核心的實踐

- XP 的核心可被分為四個範圍
  - 小規模回饋 Fine scale feedback
    - 測試驅動開發、策劃遊戲、客戶、成對程式設計
  - 反覆持續性程序 Continuous process
    - 持續整合、設計最佳化 (軟體重構) 、小型發佈
  - 共識 (標準與規章) Shared understanding
    - 簡單設計、集體程式碼所有、程式設計標準 / 規約
  - 程式設計者的福利 Programmer welfare
    - 穩定標準的速率

小規模回饋 Fine scale feedback

- 測試
  - 沒有經過測試的程式碼什麼都不是
  - XP 認為，如果一個函數沒有經過測試就不能認為它可以工作
  - 單元測試
    - 用以測試一小段程式碼的自動測試。在 XP 中，需要在程式碼編輯前就編輯單元測試。這種方式的目的是激勵 Programmer 思考自己的 Code 在何種條件下會出錯。 XP 認為當 Programmer 無法再想出更多能使 Code 出錯的情況時，這些程式碼便算完成。

小規模回饋 Fine scale feedback

- 成對設計
  - 它迫使我們與別人溝通、把別人跟自己的想法看得更清楚、也加快了寫程式的速度，對新手而言（每個人都是某方面的新手），這更是一條學習的捷徑，大家都知道，有人當場教，學得最快。
  - 一個 programmer 控制電腦並且主要考慮編碼細節。另一個主要注意整體結構，不斷的對第一個 programmer 寫的程式碼進行反饋。
  - 成對的方式不是固定的 :
    - XP 甚至建議程式設計師盡量交叉結對。這樣，每個人都可以知道其它人的工作，每個人都對整個系統熟悉，成對程式設計加強了團隊內的溝通。

小規模回饋 Fine scale feedback

- 客戶 (現場客戶)
  - 在 XP 中，「客戶」並不是為系統付帳的人，而是真正使用該系統的人。
  - 極致編程認為客戶應該時刻在現場解決問題。
    - 例如：在團隊開發一個財務管理系統時，開發小組內應包含一位財務管理人員
  - 一個小組理論上需要一個 User 在身邊 , 制定軟體的工作需求和優先等級，並且能在問題出現的時候馬上回答 (實際工作中，這個角色是由客戶代理商完成的)

小規模回饋 Fine scale feedback

- 策略遊戲
  - 策劃程序分為兩部分
    - 發佈策劃
    - 反覆狀態

策略遊戲

- 發佈策略
  - 照價值排序：業務人員按照商業價值為使用者需求排序。
  - 按風險排序：開發者按風險為使用者需求排序。
  - 設定周轉率：開發者決定以怎樣的速度開展專案。
  - 選擇範圍：挑選在下一個版本 (工作點) 中需要被完成的使用者需求，基於使用者需求決定發佈日期。
- 值得一提的是
  - 在作業階段開發人員和業務人員可以「操縱」整個程序。意指他們可以做出改變。某個使用者的需求，或是不同使用者需求間相對優先等級，都有可能改變；預估時間也可能出現誤差。也就是說這是在整個策略中，准許做出相應調整的機會。

策略遊戲

- 反覆狀態 (計畫)- 探索階段
  - 關於建立任務和預估實施時間。
    - 收集使用者需求：
      - 收集並編輯下一個發布版本的所有使用者需求。
    - 組合 / 分割任務：
      - 如果程式設計師因為任務太大或太小而不能預估任務完成時間，則需要組合或分割此任務。
    - 預估任務：
      - 預測需要實作此任務的時間。

策略遊戲

- 反覆狀態 (計畫)- 約定階段
  - 在約定階段以不同需求作為參考的任務被指派到 Programmer
    - 接受任務：
      - 每個 Programmer 都挑選一個所需負責的任務。
    - 預估任務：
      - 由於 Programmer 對此任務負責，所以必須給出一個完成任務的估計時間。
    - 設定負載係數：
      - 負載係數表示每個 Programmer 在一個反覆中理想的開發時間。
      - 一周工作 40 小時，其中 5 小時用於開會，則負載係數不會超過 35 小時。

策略遊戲

- 反覆狀態 (計畫)- 約定階段 2
  - 平衡：當團隊中所有 Programmer 都已經被配置了任務，便會在預估時間和負載係數間做出比較。任務配置在 Programmer 中達到平衡。如果有 Programmer 的開發任務過重，其它 Programmer 必須接手他 / 她的一部分任務，反之亦然。

策略遊戲

- 反覆狀態 (計畫)- 作業階段
  - 各個任務是在反覆計劃的作業階段中一步步實作的。
    - 取得一張任務卡片：
      - Programmer 取得一張由自己負責的任務的卡片。
    - 找尋一名同伴：
      - 這個 Programmer 將和另一位 Programmer 一同完成開發工作，也就是成對設計。
    - 設計這個任務：
      - 如果需要，兩位 Programmer 會設計這個任務所達成的功能。

策略遊戲

- 反覆狀態 (計畫)- 作業階段 2
  - 編輯單元測試：
    - 在 Programmer 開始編輯實作功能的程式碼之前，他 / 她們首先編輯自動測試。
  - 編輯程式碼：
    - 兩位 Programmer 開始 Coding 。
  - 執行測試：
    - 執行單元測試來確定程式碼能正常工作。
  - 執行功能測試：
    - 執行功能測試 (基於相關使用者需求和任務卡片中的需求) 。

反覆持續性程序 Continuous process

- 持續整合
  - 我們都知道開發工作是需要使用最新的版本同步開發
  - 每一個 Programmer 都會有做出不同的改變或新功能的添加，在他們個人的電腦中
  - XP 要求每個人需要在固定的短時間內上傳，或者是每當發現重大錯誤，或成功時上傳。
  - 這可以有效避免，因為 cycle 導致的 delay 或 waste

反覆持續性程序 Continuous process

- 軟體重構
  - 由於 XP 提倡 Coding 時只滿足目前的需求，並且以盡可能簡單的方式實作。
  - 有時會碰上一套僵硬的系統，所謂僵硬的系統， XP 教條認為當這種情況發生時，意味著系統正告訴你通過改變系統架構以重構程式碼，使它更簡單、更泛用。
  - 僵硬系統 - 需要雙重（或多重）維護：
    - 功能變化需要對多份同樣（或類似）的程式碼進行修改；或者對程式碼的一部分進行修改時會影響其它很多部分。

反覆持續性程序 Continuous process

- 小型發佈
  - 把整個專案分成好幾的段落，進行 release ， 在 project 開始就決定，何時要 release 個各段落。
  - 這樣的小型發佈，可以增進客戶對整個 project 的了解與信心
  - 這些小型發佈，只是測試版，並不會繼續存在
  - 這也使得 User 可以在各部份提出自己的意見

共識　Shared understanding

- 簡單設計
  - 第一個觀念“簡單是最好的”
  - 每當一個新的段落被完成， Programmer 必須思考，是不是有更簡單的方法可以達成同樣目的
  - 如果是，則必須要選擇比較簡單的方法
  - 軟體重構也是用來完成，簡單化的工作

共識　Shared understanding

- 集體所有制
  - 集體所有制表示每個人都對所有的程式碼負責；反過來又意味著每個人都可以更改程式碼的任意部分。成對程式設計對這一實踐貢獻良多：藉由在不同的 pair 中工作，所有的 programmer 都能看到完全的 code 。
  - 集體所有制的一個主要優勢是提升了開發程序的速度，因為一旦程式碼中出現錯誤，任何人都能修正它。
  - 當然在給予每個人修改程式碼的權限的情況下，可能存在 Programmer 引入錯誤的風險，他 / 她們知道自己在做什麼，卻無法預見某些依賴關係。完善的單元測試可以解決這個問題：如果未被預見的依賴產生了錯誤，那麼當單元測試執行時，它必定會失敗。

共識　Shared understanding

- 程式設計標準 Standard
  - 在開始之前，整個團隊必須制定且同意一些標準的規則
    - Code 的格式、 Language 的選擇、客戶的要求 ... 等

穩定標準的速率 Programmer welfare

- 又稱“四十個工時”，這種聽起來像『大老闆的慈悲』的規則， XP 認為應該內建成為一種工作倫理，準時下班是一種美德。畢竟，如果家庭沒有了、身體健康沒有了，就算寫出『宇宙無敵』的軟體，大概也找不到人分享，很多書籍的作者，不都把書獻給家人嗎？
- 它的實現，重點在於完整詳細的計畫，在 project 剛開始，就決定好要在這個 release version 達成什麼樣的目標。

爭論的觀點

- XP 也有其被爭論的一面：
  - 沒有書面的詳細的規格說明書
  - 客戶代表被安排在專案中
  - 程式設計師以 pair 的方式工作
  - 測試驅動的開發
    - 絕大多數設計活動都匆匆而過，並且是漸進式的，從一個「最簡單的可能工作單位」開始並只有在其需要時（測試失敗）才增加複雜性。
    - 只依靠單元測試促成為了設計紀律。

總結

- XP 針對的是中小型團隊和中小型專案，但世界上畢竟還有大型專案跟超大型專案，究竟這種重視人甚於重視製程方法論，能不能跟其他重視製程的方法論分庭抗禮呢？
- 當然 XP，還需要時間來證明它自己！
- 輕量級跟重量級分別代表著兩個極端，一邊是注重人甚於製程，另一邊則是重視製程甚於人。
  這代表一個不甚有經驗的人，也可以照著重量級製程的種種規則一步步來，優點當然是穩定，缺點當然是僵化。
  輕量級製程也差不多，好處是彈性，壞處是混亂。

總結

- 所以，取長補短吧！導入任何製程都不免要做些修正，沒辦法全盤複製的。導入 XP 時，有公式用公式，沒公式用模式，如果都沒有，那就只好用常識了。

Reference

- 極致軟體製程 (書，中譯版)
  http://140.109.17.94/Jyemii/xpcolumn/xpexplained/TranslatePreface.htm
- Wikipedia- 極限製程 (中、英版)
- Wikipedia- 軟體重構
- Mingster‘s Bliki 九月 2007
  http://mingstert.blogspot.com/2007_09_01_archive.html 中 Chinese about XP programming 一文
- 新浪網 http://financenews.sina.com/sinacn/304-000-106-109/2008-03-30/0500731317.html
- 終極製程專欄
  http://tropic.iis.sinica.edu.tw/xp/XP.htm
- http://www.xprogramming.com/xpmag/whatisxp.htm