Homework 1

10910CS 546000 Software Project Management　王傳鈞　109062631

第一題

根據ISO 9001:2015的內容，此標準的目的在於有效地促使品質管理系統 (quality management system, QMS) 能符合顧客要求，而軟體專案管理 (software project management, SPM) 是專案管理 (project management, PM) 應用在軟體工程領域的一種體現；因此，SPM肯定能導入ISO 9001:2015並符合其規範；以下舉例三個不同的面向，分別說明  
ISO 9001:2015與SPM之間如何建立緊密的關係。

1. **Customer Focus與DevOps**

在ISO 9001:2015 §5.1.2 闡述客戶導向 (customer focus) 的概念：希望高階主管要保持對客戶的關注，確保顧客提出的要求都有盡量滿足；這相當類似著名軟體開發文化—DevOps。DevOps改進在傳統模式中，軟體開發 (development) 團隊只會在持續收集一陣子的使用者回饋後，方才著手統整前版本的缺失並規劃下一個版本，如此緩慢的流程將令終端用戶無法感受到良好的使用體驗。

DevOps仰賴IT運維 (operation) 團隊日常收集各式系統日誌與使用者回饋，並且持續回報給development團隊當作軟體開發方針；而development團隊打破舊有方式，拆分整個版本成為數個細小功能點的更新，每當完成一個功能點就進行軟體整合、測試、部屬至平台，也就是所謂的持續整合、持續部屬 (continuous integration, continuous deployment, CI/CD)。倘若企業能成功運行DevOps，則可讓終端用戶感受到「立刻改善」的使用者體驗，也就完全契合ISO 9001:2015的客戶導向精神。

1. **Engagement of People與Lean Software Development**

ISO 9001:2015 §7強調「全員參與」：全體員工必須了解自己在整體計畫當中的角色、高階主管要能適度地授予自主決策權給員工、團隊成員彼此需要激勵主動學習的風氣。此三個概念與精益軟體開發 (lean software development, lean) 強調的七大原則之中：optimize the whole、empower the team、amplify learning不謀而合。顯見徹底落實ISO 9001:2015的開發團隊將能輕鬆導入lean法則，減低軟體開發過程之無謂浪費，達成「如預算」之目標。

1. **Process Approach、PDCA與SDLC**

ISO 9001:2015的核心概念—過程導向 (process approach) 意味：團隊在乎輸入資源轉化成最終績效的過程是否有效率，不再端看最終績效是否達標；為此，知名的plan-do-check-action (PDCA) 方法，就很適合拿來實踐process approach的精神。根據字面上的意思，我們不難發現SPM領域知名的軟體生命週期 (software development life cycle, SDLC) 模型與PDCA幾乎在訴說相同的過程。

在SDLC當中的planning和analysis 階段，等同PDCA之plan階段；SDLC的design、implementation階段，契合PDCA之do階段；SDLC之testing階段與PDCA的check階段類似，都是針對前些階段進行仔細審核，以期抓出先前犯下的錯誤；最後的SDLC—maintenance與PDCA—action階段，都是照著前一個階段所指出的錯誤，來進行修正與改善。

第二題

* **DevOps. *IEEE Software, 33*(3), 94-100**
* **文章概要**

提到SDLC的話，一般人直覺想到的大多是歷史悠久的waterfall model；但是，在現今市場潮流急遽變化之下，消費者已經無法忍受長達半年至一年，都沒有獲得新版本的軟體更新，或是遲遲無修復現有軟體的缺陷。為了因應這樣子的潮流，有些企業開始引入敏捷軟體開發 (agile software development, agile)；藉由分割、切細軟體版本，讓每次的版本疊代 (iteration) 只需改進一小部分的功能 (break product development into small increments)，大幅縮短衝刺 (sprint) 的時間，且令使用者能不停獲得版本更新。乍看之下agile似乎已是最佳解，然而本文章點出一個agile的盲點：開發部門 (development) 一昧快步小幅度更新，忽視來自IT運維部門 (operation) 的建議與使用者的回饋 (feedback)，終究將陷入「為了更新而更新」的窘境。本文章希望透過推廣DevOps精神，讓agile團隊能夠更靈活、有效率地開發。

文章分為三大部分：首先，作者概要式介紹DevOps (page 1-2)；接著，作者順著DevOps開發三階段—build-deployment-operation，分別介紹不同階段有哪些知名的套件、環境可供使用 (page 2-5)；最後，文章分析真實案例AWS平台如何整合DevOps，以及第一作者推銷自己開設的公司Vector Consulting Services提供什麼服務 (page 5-7)。

綜觀全文，我認為最精華的部分屬「第一部分：簡介DevOps」，可惜的是此處篇幅不多，反倒是本文章之參考文獻Understanding DevOps & Bridging the Gap from Continuous Integration to Continuous Delivery [1]更值得仔細閱讀。建議對於DevOps的細節有興趣的讀者可直接閱讀該文章，待稍後才回過頭閱讀本文章的其他部分，認識現行的知名軟體與環境。

* **心得與體悟**

由於以前完全沒有接觸過軟體工程領域，因此對於相關的專有名詞與其定義並不熟稔。趁著閱讀這篇文章的機會，迫使我必須額外查詢、閱讀其他的參考文獻，對於agile、scrum、  
lean SD、XP、V-Model、test-driven development、DevOps、CI/CD等名詞有初步的認識。另外，在閱讀的過程中，也讀到許多軟體測試領域的術語（例如：smoke test, sanity test, white-box/ black-box testing等），這部分暫且不深入探究，留待日後建立完整的軟體度量觀念，才搜尋相關文獻來閱讀。

References

1. Manish Virmani (2015, May). *Understanding DevOps & Bridging the Gap from Continuous Integration to Continuous Delivery.* Paper presented at the INTECH 2015.