壹、緒論

軟體工程的理念被提出至今，已將近50年，這段期間許多學者與專家不斷為提升軟體品質與開發生產力而努力，持續推出新的開發方法與工具，用以解決與改善軟體危機的困擾（Pressman, 2010）。瀑布模式（Waterfall Model）於1970年代被提出（Schach, 2011），延續至今已有一段時間，仍是知名企業與組織開發大型軟體系統的最愛。瀑布模式對於各個階段的文件要求特別嚴謹，除了須具備高度的完整性與正確性外，尚必須通過正式審核作業，才能進入下個開發階段。此外，改良的瀑布模式也融入許多彈性與特質，於階段審查作業發現的問題與缺失，可以適時回饋至相關階段，進行及時的修訂與改正，避免問題與缺失持續擴大。不過，瀑布模式不允許反覆且漸進的需求規格，也就是所有的需求項目必須被一次提出，開發作業才能進入設計階段，因而導致少數核心需求存在情節描述不清、狀態定義不明確且缺乏可變動性等缺失，大幅增加軟體大型專案的失敗風險。

Standish Group是一家探究軟體開發專案的知名顧問公司，依據該公司對9,236件軟體專案的剖析報告指出，只有29%的軟體專案是成功的，18%的軟體專案於執行過程中被取消，而剩餘的53%是屬於延後完成、超出預算或是產品未能滿足需求等狀況（Hayes, 2004）。探究軟體專案失敗的原因都與需求密切相關，需求規格是軟體系統開發作業的依據，但部分核心需求存在情節描述不清、介面定義不正確或不一致等缺失，將無法達成溝通且建立使用單位、開發團隊及利害關係人之間的共識，大幅衝擊計畫正常且有效地運作。軟體開發過程中，專案必須面臨環境變遷與需求變更的挑戰（Mantel, Meredith, Shafer, and Sutton, 2011），核心需求缺乏可變動性，將使得每次環境變遷或需求變更就拉開需求與開發作業之間的距離，漸漸導致專案計畫完全無法控管軟體開發的運作，間接導致專案的失敗（賴森堂，2012；Boehm, 1991; Weller, 1994）。需求是產品確認與系統驗證的基礎，需求缺乏可確認特性，將使系統無法通過驗收測試，成為失敗的專案。這些衝擊專案成敗的核心需求項目即為關鍵需求項目（Critical Requirement Items, CRI）。為了降低專案的失敗風險，CRI應融入溝通、確認與變動等品質，以優勢的品質克服各種開發作業的挑戰。

行為驅動開發（Behavior Driven Development, BDD）（North, 2006）是一套適合反覆與漸增開發（Iterative and Incremental Development, IID）方法（Larmnan and Basili, 2004）的軟體製程，BDD具備下面幾項特性：一、在軟體開發過程中，使

用者特定需求（User Stories）的測試個案可以事先規劃與設計（Cohn, 2004; Wake, 2012）。二、協助建立最終使用者、開發人員與利害關係人之間的共識。三、提供自動化的迴歸測試（Regression Testing）。四、具備輔助行為描繪內容的修訂與設計重構（Refactor）的能力。BDD的特性足以協助改善CRI欠缺的品質，以BDD製程識別CRI的品質缺失且協助品質改善作業，系統需求品質可以因CRI品質改善而提升，進而有效降低軟體專案失敗風險。本文討論影響需求品質的CRI，將BDD製程融入CRI的品質改善作業，且以BDD與品質量測模式為基礎，規劃一套CRI品質改善機制，用以評估、標示與改善CRI品質缺失，適時地提升CRI溝通、確認與變動等品質特性，具體降低軟體專案開發風險。本文第貳節針對傳統開發模式的優劣與BDD的運作方式進行探討。第參節剖析影響軟體專案成敗的CRI，且深入討論以BDD製程優勢識別CRI 的品質缺失。為了具體改善CRI品質缺失，第肆節規劃一套以BDD與品質量測模式為基礎的CRI品質改善機制，及時標示出CRI品質缺失，並適時地提出矯正措施以改善CRI品質。以選課子系統為例，第伍節針對CRI品質改善機制的效益進行評估。第陸節再次強調CRI品質的重要性，說明CRI改善機制的優勢未來方向，且針對本主題做出結論。

貳、傳統開發模式與BDD之探討

制度完善的企業與組織選擇傳統且保守的開發模式開發大型軟體系統，本節探討傳統軟體開發模式的優劣及適用於IID方法的BDD製程。

一、傳統開發模式的優劣

軟體開發方法隨著資訊技術、運作環境與使用者要求等方面的成長，不斷的演進中，從最早期的Fix-code到近期的MDA（Model Driven Architecture）、敏捷開發模式（Agile Process）等（吳仁和、林信惠，2013；Schach, 2011），多達十幾種開發模式，大部分的開發模式都與使用者需求有著密切的關係（Schach, 2011; Sommerville, 2010）。瀑布開發模式對於需求的遵循性要求很高，未能完整提出且擬訂明確的需求規格，軟體開發作業就不能進入下一階段。近期提出的開發模式（如同步（Concurrent）開發模式、RUP（Rational Unified Process））（Schach, 2011）、敏捷開發模式已經改變使用者需求制訂的方式，不再要求使用單位必須一次提出完整且明確的使用者需求，反而配合反覆的開發作業採取漸進方式提出需