以量測模式建立優質的類別關係之研究
A Study of Applying Measurement Model for Creating High Quality Class Relationships

賴森堂
實踐大學 資訊科技與管理學系
stlai@mail.usc.edu.tw

摘要

物件導向分析與設計階段所規劃出的類別元件與類別關係不僅是構成物件導向軟體系統的基本要件，更是後續開發、測試與維護等階段作業的重要依據。正確、完整、一致且具統一格式的類別設計文件是確保品質的基本要求，類別關係的關鍵品質更是提昇後續作業品質的必要條件，卻因缺乏一套具體且有效的審核與改善機制，經常造成類別關係品質缺失無法及時排除，導致後續開發作業受到極大影響。為此，本文深入探討類別關係品質缺失對於後續作業的衝擊，且配合類別關係檢視活動的量化程序，提出一套類別關係品質量測(Class Relationships Quality Measurement; CRQM)模式，以量化機制，協助識別類別關係存在的品質缺失，且衍生出問題矯正與缺失改善方法，具體且有效的提昇類別關係的設計品質。

關鍵詞：類別設計、類別關係、品質量測模式、檢視活動、品質因子。

1.緒論

軟體開發製程，由於開發人員的疏失所注入的錯誤與缺失，對軟體產品品質將產生一定的衝擊，錯誤與缺失存在的時間越久，造成日後修正成本越高且對於系統的影響範圍越大[18]。設計階段疏失造成的錯誤與缺失，進行修正的成本是需求階段的許多倍，設計階段隱含的錯誤與缺失未能及時修正，將遞延至後續階段且持續擴大，造成後續的修訂成本大幅的增加[18][19]。因此，軟體發展的過程中，若不能在早期階段中有效的減少錯誤產生，將導致軟體品質降低，開發時程延誤，且使得後續的測試及維護階段成本大幅提高。物件導向開發過程中的每一個階段，都必須引用並且參考前面階段所產生的文件，以進行階段性開發作業。物件導向設計階段引用物件導向分析所完成的文件，進行軟體系統架構及相關元件組成關係的描繪作業。物件導向設計所規劃出的類別圖(Class Diagram)是設計階段與程式製作之間的溝通媒介，更是程式設計、測試及維護等相關人員必須引用的關鍵性文件[3][6]，因此如何產生正確、完整且具高品質的類別設計文件，提供後續階段開發人員一套可以依循的方向，是類別設計作業必須重視的議題。物件導向設計階段所完成的類別圖，不僅是後續開發、測試與維護作業的依據，對於後續開發作業與整個軟體品質更有相當關鍵的影響力。

為了提昇物件導向設計的品質，相關的研究陸續被提出，包括 Alloy[14]、Syntropy[8]、Catalysis[10]與 OCL[17]等，其中物件限制語言(Object Constraint Language; OCL)是一套 OMG 所制定及維護的形式語言(Formal Language)，主要用來描述以 UML 塑模過程中，物件本身及物件與物件之間的操作限制，透過 OCL 可以排除原有 UML 使用自然語言描述時所導致的不確定性，並以 OCL 所訂定的限制條件彌補 UML 類別圖繪製上的不足，大幅提昇類別設計的明確性與完整性。不過，類別關係是發揮物件導向特性的關鍵，OCL 對於類別關係的限制與品質要求卻沒有明確的規範，使得類別圖中的類別關係將遭受限制問題與品質缺失的困擾，進而影響系統後續開發作業的效率與品質。

類別圖是物件導向設計的指標性產品，主要是用來描述軟體系統的靜態觀點，

除了可以展現系統的資訊結構，更可以配合其他模式圖描述出系統的動態行為，因此類別圖在 UML 塑模中成為其他模式圖的引用依據[6]。類別圖中定義的類別若要發揮較大效用，就必須與其他類別充分合作，類別之間的合作方式是建立在四項關係上，四項關係包括相依關係(Dependence)、一般化關係(Generalization)、關聯關係(Association)、實現化關係(Realization)等[3]。類別之間的結合關係是發揮物件導向特性的關鍵，不過，類別關係(Class Relationships)一旦缺乏一致性、模組性或適度的限制，將對系統後續的開發、測試與維護作業帶來難以克服的衝擊[4][13]。本文深入探討類別關係出現的問題與品質缺失，及其對於後續作業造成的影響，進而以檢視為基礎，提出一套類別關係品質量測 *(Class Relationships Quality Measurement; CRQM)* 模式，以量化的檢視方式，協助標示出類別關係的品質問題與缺失，協助及時矯正問題與改善缺失，具體提昇類別設計品質。本文第二節將說明類別設計在開發過程中扮演的角色，以及類別設計所規劃與定義的類別關係。改善類別關係品質是提昇類別設計品質的關鍵，第三節將探討量化與量測類別關係品質的前置作業，且針對蒐集與正規化類別關係品質因子進行描繪。第四節將以檢視活動為基礎，提出一套類別關係品質量測(*CRQM)*模式，且說明配合量測模式的改善作業。第五節將再次強調類別關係的重要性，且以 CRQM 模式的優勢說明本研究的貢獻。

2. 類別設計的重要性

類別設計是物件導向設計階段與程式製作階段之間的溝通媒介，具備一致性與關鍵品質的類別設計是確保軟體系統品質的必要條件。

2.1 類別設計扮演的角色

軟體開發過程中，需求分析階段與設計階段所產生的錯誤率，占整個軟體開發階段的 85%[5]，設計過程所注入的錯誤與缺失若不能及時更正與改善，將遞延至後續階段且持續擴大，不僅將衝擊階段性的產品品質，更使得後續的測試及維護階段開發成本大幅提高。物件導軟體開發過程中，各個階段都有其特定的任務須達成，各項工作都必須引用或參考前面階段所完成的產品文件，以進行階段性的開發作業[19]。物件導向設計依據物件導向分析的成果，描繪軟體系統的架構及相關元件的組構關係，物件導向設計所規劃出的類別圖是設計階段與程式製作之間的溝通媒介，更是程式設計、測試及維護等階段必須引用的關鍵文件，因此如何產生完整且一致的類別設計文件，提供後續階段開發人員一套可以依循的方向，是類別設計作業必須重視的議題，類別設計是物件導向設計階段的指標性產品，對於整個軟體系統品質及後續開發作業有著高度的影響力。同時，在 UML 塑模過程中，類別設計完成的類別圖也具有承先啟後的重要任務[16]，如圖 1 所示，類別設計與類別圖是軟體開發作業的核心項目。

物件導向分析後，開發作業進入物件導向設計階段，此階段涵蓋兩項主要活動：架構設計 (Architectural design) 及細部設計 (Detailed design)[18]。架構設計的主要工作是規劃軟體的整體架構並將軟體進行模組化的分解，在物件導向模式中，部分的架構設計，可以在物件導向分析階段中進行[1][2]。架構設計所規劃出的類別元件與類別關係是構成物件導向軟體系統的基本要件，完成架構設計且進入細部設計階段後，必須針對每一個類別元件進行詳細的定義與描述，以完成具備高品質的類別設計文件，引用且參考這些類別設計文件，程式設計人員方能撰寫出高品質的程式[2]。基於物件導向的設計原理，類別設計可以劃分成類別架構設計與類別細部設計兩個步驟，描述如下[2][3]：

(1) 類別架構設計包括類別元件篩選與類別關係設計等步驟：

- 類別元件篩選：透過已完成的使用個案圖與使用個案情節描述內容，初步的識別出候選的類別，再從候選的類別中篩

選適當且必要的類別元件。確認後的類別元件必須依其用途給予類別唯一且有意義的命名，且明確定義類別元件本身的屬性(資料成員)與行為(成員方法)。

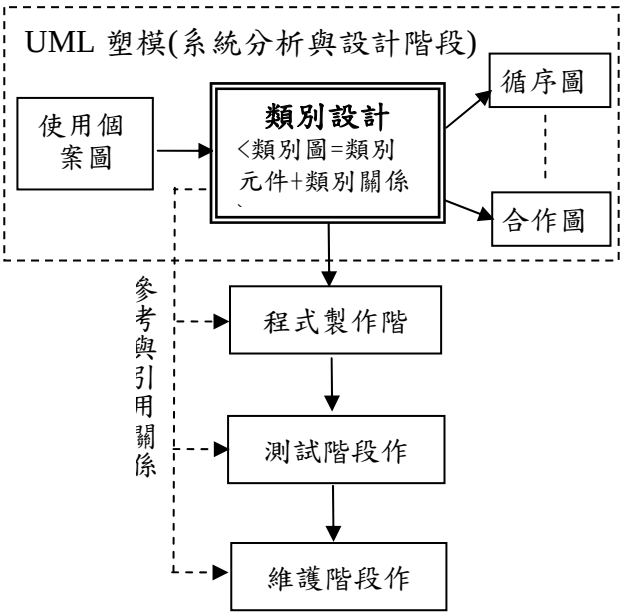

圖 1 類別設計與前後階段的關係圖

- 類別關係設計：類別圖是物件導向架構設計的主要產品之一。在類別圖中，將明確定義類別元件之間的相依、一般化、關聯與實現化等關係，一套設計完善的類別關係可以具體提昇產品的生產力(Productivity)及可維護性(Maintainability)，反之，類別關係一旦設計不當，可能導致模組性(Modularity)、可測試性(Testability)與可維護性降低，複雜度(Complexity)提高，且缺乏修改彈性(Flexibility)等重大品質缺失。

(2) 類別細部設計包括類別介面、成員方法和資料成員、細部邏輯結構及細部資料結構等步驟：

- 類別介面設計：類似於結構化設計(Structured design)的模組關係，類別元件之間的呼叫關係，必須在類別元件設計中進行詳細定義。參數傳遞和呼叫層級的設計，是類別介面設計的主要任務。
- 成員方法(Member methods)和資料成員(Data members)設計：具封裝性的類別元件是由一些成員函數和資料成員所組構而成的。為了產生高品質的類別設計規格，此項設計的主要任務必須明確的定義和具體說明類別元件的成員函數和資料成員。
- 細部邏輯結構設計：成員方法定義完成之後，成員方法的細部邏輯也必須被進一步的描述。在此項設計過程中，該成員函數的細部邏輯可藉由程式設計語言(Program Design Language, PDL)或虛擬碼(Pseudo code)進行清楚且具體說明[18][19]。
- 細部資料結構設計：資料成員定義完成之後，共同的資料成員及成員函數個別引用的資料結構也必須被詳細描述。在此項設計任務中，類別元件中成員函數的細部資料結構可藉由變數型態、範圍及初始值等定義進行清楚且具體說明。

類別設計的階段性產品，對於後續階段的開發有很大的影響，包括程式製作階段、整合測試階段及維護階段等，彼此之間都與類別架構及類別細部設計有緊密的關連，類別設計文件與後續開發階段之間的相對關係如同表 1 所示，因此具備高品質的類別設計文件，才能減少錯誤遞延至後續的開發階段，而具體提昇後續開發階段的品質與生產力。

2.2 類別關係對於後續作業的影響力

類別關係是發揮物件導向特性的關鍵，不過，類別關係一旦缺乏一致性、模組性或適度的限制，將對系統後續的開發與維護作業帶來難以克服的衝擊。類別設計定義的類別元件要充分發揮物件導向的特性，就必須與其他類別充分合作，強化更完整的整合能力，類別之間的合作方式是建立在相依、一般化、關聯及實現化等四種結合關係上，這四種結合關係的特質及具備的特性說明如下:

表 1 類別設計與後續開發階段的對映關係表

設計子工作項 \ 後續開發階段		程式設計階段	整合階段	維護階段	
				擴充	修改
類別架構設計	類別元件篩選	✓	✓	✓	✓
	類別關係設計	✓	✓	✓	✓
類別細部設計	介面設計	✓	✓	✓	✓
	類別成員方法與資料設計	✓		✓	✓
	類別邏輯結構設計	✓			✓
	類別資料結構設計	✓			✓

(1) 相依關係：是一種類別元件之間的使用關係，表達一個類別元件可以使用其他類別元件所定義的資源，且引用的類別元件可能會受到被用類別元件改變的影響，被用類別元件則不受影響。設計完善的類別相依關係可以提高開發的生產力，更可以將可再用性(Reusability)、模組性、可測試性、可維護性與整合性等特性融入類別設計中。

(2) 一般化關係：類別元件的一般化關係就是類別元件之間的繼承 (Inheritance)特性，有時被稱為「is-a-kind-of」的關係，反向的特性即是特殊化(Specialization)。一個類別元件可以繼承一個以上的父類別，子類別繼承父類別，就可以使用父類別所定義的成員方法與資料成員，一般化關係將可再用性、模組性與整合性(Integrity)融入類別設計中，有效提昇生產力與可維護性。不過，類別元件繼承或被繼承的層次過多，或繼承後的覆蓋情形過於浮濫都會造成不利的反效果。設計完善的類別一般化關係可以提高開發效率，更可以將可再用性、模組性與可維護性等特性融入類別設計中。

(3) 關聯關係：是一種類別元件之間的靜態結構關係，描述類別與類別間之連結。物件之間若存在某種鏈結關係，表示物件各自所歸屬的類別也會存在某種關聯，意味著某一類別之物件知道另一類別物件的存在。在類別圖的關聯關係中可以使用四種標記(Notation)：關聯名稱、角色名稱、多重性(Multiplicity)與可通性(Navigability)來輔助描述。設計完善的類別關聯關係可以將模組性、可測試性、可維護性與整合性融入類別設計中。聚合關係(Aggregation)與組合關係(Composition)是關聯關係中較為特殊的關係，其定義與說明如下：

- 聚合關係：代表一組主類別(Main Classes)依賴副類別(Sub-Class)的關係，當副類別不存在，主類別也無法存在，副類別是否存在則不受主類別的影響。
- 組合關係：代表一組主類別(Main Classes)依賴副類別(Sub-Class)的關係，當副類別不存在，主類別也無法存在；同樣地，當主類別不存在，副類別也無法存在。

(4) 實現化關係：實現化關係表達一類別之行為是由另一類別來描述。物件導向技術中允許一類別可以實作其他類別定義之項目，不過該實作類別必須遵守介面的(Interface)定義，可以不需要動用繼承關係。設計完善的實現化關係可以將模組性、可測試性與可維護性融入類別設計中。

透過類別元件之間的結合關係可以於設計階段融入多項關鍵品質特性，這些品質特性可以有效提昇後續開發的效率與維護作業的品質，類別結合關係融入的品質特性如表 2 所示。

表 2 類別結合關係融入的品質特性彙整表

類別關係 \ 品質特性	可再用性	模組性	可測試性	可維護性	整合性
相依關係	V	V	V	V	V
一般化關係	V	V		V	
關聯關係		V	V	V	V
實現化關係		V	V	V	

3. 類別關係品質因子蒐集與正規化

量化類別關係品質是改善類別設計品質的關鍵，蒐集與正規化類別關係品質因子，是量化與量測類別關係品質的前置作業。

3.1 類別關係的關鍵品質

類別之間建立的關係缺乏一致性或不能配合適度的限制條件，將對系統後續的開發與維護作業帶來難以克服的衝擊。以下針對四項結合關係的基本品質與限制性要求的衝擊進行探討與說明，且將品質缺失與限制性問題對後續作業的影響，加以匯整如表3所示：

(1) 相依關係的問題與衝擊：
- 相依關係的一致性：類別成員方法之間的相互叫用關係及類別之間的資料成員間引用關係若出現不一致，將不利後續的開發、測試與維護等作業。
- 相依關係的耦合度：類別成員方法叫用其他類別成員方法應該適度採用資料或資料結構的溝通介面，有效降低耦合度，以避免影響後續的測試作業與維護品質。
- 相依關係引用限制：類別引用成員方法其他類別的成員方法或資料成員應該適度的限制，否則將增加關係的複雜度，影響後續開發、測試與維護等作業的品質。
- 相依關係被引用限制：類別的成員方法或資料成員被其他類別引用應該適度的限制，否則將增加關係的複雜度，影響後續開發、測試與維護等作業的品質。

(2) 一般化關係的問題與衝擊：
- 一般化關係的一致性：類別與類別間的繼承與被繼承關係若出現不一致，將不利後續的開發、測試與維護等作業。
- 一般化關係的深度(deepness)：類別被太多層次子類別所繼承或是同時繼承過多層次的祖先類別都將造成後續開發與維護作業的困擾，對於類別一般化關係的層次應該適度限制其深度。
- 一般化關係的廣度(breadth)：類別被太多子類別同時繼承或是同時繼承過多的父類別都將造成後續開發與維護作業的困擾，對於類別一般化關係應該適度的限制其廣度。
- 一般化關係的覆蓋度：父類別成員方法與資料成員被覆蓋行為應該有一定的規範，才具體發揮繼承的特質，一般化關係若不能有效限制覆蓋行，不僅將影響類別間的透視性，對於可測試性、可再用性及可維護性等品質也將大幅滑落。

(3) 關聯關係的問題與衝擊：
- 關聯關係的一致性：類別元件之間的結構定義關係若出現不一致，將不利後續於衍生物件後的運作行為。
- 關聯關係的完整性：具備關聯特性的類別必須建立完整的結合關係，才能配合資料庫架構設計有效提昇資料存取的效率。
- 關聯關係的複雜度：一個類別與其他類別間的結構定義關係應該有一定的規範，以適度的限制類別元件之間的關聯關係引用，否則將影響後續的開發、測試與維護等作業品質。

(4) 實現化關係的問題與衝擊：
- 實現化關係的一致性：類別元件與介面之間的實作關係必須存在一致性的定義，且可以明確與完整的產生交互引用對照表，實現化關係若出現不一致現象，將不利後續的測試與維護作業，且降低抽象化的用意。
- 實現化關係引用限制：類別元件實作的介面個數應該適度的限制，以

有效控制類別元件與介面之間的耦合度，降低測試與維護作業的複雜度[4]。

- 實現化關係被引用限制：介面被類別元件實作的個數應該適度的限制，以控制類別元件的耦合度，有效控管介面實作後的應用與品質，降低其變動的衝擊度。
- 實現化關係抽象化：介面定義的屬性與行為必須適度的限制，應過多且複雜的屬性與行為勢必會衍生出後續修改、測試與維護作業的困擾。

表 3 類別關係問題與不一致衝擊彙整表

受影響作業 / 類別關係	程式製作	單元測試	整合測試	維護作業
相依關係	V		V	V
一般化關係	V	V		V
關聯關係	V		V	V
實現化關係	V	V		V

3.2 以檢視作業蒐集量化品質因子

許多 UML 的開發工具可以協助找出類別設計的不一致現象，且透過錯誤訊息協助設計人員及時修改錯誤，提高類別設計的一致性，不過，類別設計的模組性、複雜度及限制性等特質，並非正確或錯誤的判斷，而是需要以量化的方式判斷其優劣，一般制式的開發工具不易判斷這些特質的優劣，必須經由正式的檢視作業配合品質因子的蒐集[9][11]，才能達到類別設計特質的量化效果。以下針對四項類別關係的檢視作業及關鍵品質因子蒐集方式進行說明：

(1) 相依關係的檢視項目：
- 類別的屬性或行為必須清楚定義被那些類別所引用，引用的類別也必須清楚定義引用那些類別的屬性或行為，這些引用與被引用的關係不應該出現不一致的現象。
- 不同類別的成員方法或資料成員相互叫用關係，採用的訊息傳送介面應該屬於低耦合度的資料傳送。
- 類別成員方法叫用其他類別的成員方法，必須規範適當的次數限制，且叫用次數應該在適當的次數限制內。
- 類別成員方法被其他類別的成員方法叫用，必須規範適當的次數限制，且被叫用次數應該在適當的次數限制內。

(2) 一般化關係的檢視項目：
- 父類別必須清楚定義被那些子類別所繼承，子類別也必須清楚定義繼承那些父類別，這些繼承與被繼承的關係不應該出現不一致的現象。
- 類別與類別之間的繼承深度必須規範適度的層次限制，且被繼承的父類別應該要管制被繼承的深度。
- 類別與類別之間的繼承廣度必須規範適度的層次限制，且被繼承的父類別應該要管制被繼承的廣度。
- 父類別的成員方法或資料成員被覆蓋行為必須規範適度的限制，且子類別應該依循規範的要求處理覆蓋行為。

(3) 關聯關係的檢視項目：
- 主類別與副類別之間的聚合或組合的相互關係必須被清楚定義，且定義的關係不應該出現不一致的現象。
- 類別與類別之間的關聯關係定義必須達成完整性的要求，類別關聯關係不應該出現定義不完整的現象。
- 類別與其他類別之間的結構關係定義次數必須規範適度的限制，且類別的結構關係定義次數應該接受管制。

(4) 實現化關係的檢視項目：
- 介面必須清楚定義被那些子類別所實現，類別也必須清楚定義實現那些介面，這些實現與被實現的關係不應該出現不一致的現象。

- 類別元件實作介面的個數必須規範適度的限制，且類別實作介面的個數應該接受管制。
- 介面被類別元件實作的個數必須規範適度的限制，且介面被類別元件實作的個數應該接受管制。
- 介面抽象化的屬性與行為的數量範圍必須規範適度的限制，介面抽象化的屬性與行為的數量範圍應該接受管制。

3.3 關鍵品質因子的正規化

個別或低階的品質因子無法具體表現其影響力，必須將個別或低階的品質因子適度的結合才能展現其影響力，因此高階且具體的品質項目量測值都是由一些低階的因子結合而成[5][7][15]，其中每個查核項目的評分就相當於一低階品質因子量度值，這些低階量度值依其影響度及發生率，又可設定成不同的參數或權位值，接著再配合特定的結合公式就可計算出高階品質項目的量測值。利用分析工具所蒐集到的品質因子量度值，也可以依同樣的方式計算出特定品質項目的量測值，因為個別的量度只能評量某項品質的單項特質，為了量測整體品質特性，必須將個別的量度做適當的結合。量度結合的方式可以分為線性結合(Linear Combination)[7][12][15]與非線性結合(Nonlinear Combination) [5][7][12]，其中線性結合的特質包括：思維型態較容易被接受、公式化的步驟較單純、具備較高的調整彈性及適合資料蒐集量較少之應用等。非線性結合的特質包括：能夠較精確的表現量度結合的效果、公式化的步驟較複雜且需要工具的支援、需要資深的統計技術人員的協助及適合資料蒐集量較少之應用等，線性結合與非線性結合主要特質比較內容請參閱表4。Boehm 所提出的 *COCOMO* 成本估算模式[5] 就是採用非線性結合，太複雜是使用過此模式人員的共同感受。量化品質特性是持續改善軟體品質的關鍵作業，考量量度結合方的實用性、修改彈性與簡單性等特質，本文選擇以線性結合方式來建立品質量測模式。不同層面的品質因子有不同的量度表達方式，因此在進行線性結合之前，必須將相關的品質因子量度值正規化(Normalization)，正規化的作業必須考量品質因子的發生機率、影響程度及蒐集到的量度值，優質的品質因子量度值正規化後趨近於1，劣質的品質因子量度值正規化後趨近於0，再配合事先設定的權位值及結合公式可以將低階且具高度相關性的量度結合成基層及高階的特性項目量測值。

表 4 線性結合與非線性結合主要特質比較表

量度結合模式 / 特質	線性結合	非線性結合
精確度	中等	高
公式化	步驟單純	步驟複雜
配合人員	資深軟體開發人員	資深統計技術人員
資料蒐集量	少	多

4. 類別關係的品質改善作業

本節將提出一套類別關係品質量測模式，在類別關係設計中，協助找出品質的問題與缺失，進而衍生出修正及改善等作業，以具體提昇類別設計品質。

4.1 類別關係品質量測模式

影響類別關係品質的四個關鍵分別為類別相依關係、類別一般化關係、類別關聯關係及類別實現化關係等，各個關鍵品質的影響指標則是由一些低階的品質因子所組成，透過線性結合公式，可以將高度相關性的基層品質因子結合成基層品質量度，這些基層品質量度可以進一步結合成高階項目量測值，最後再將高階項目量測值加以結合，而成為類別關係品質量測指標，以下分四個步驟說明：

(1) 類別相依關係品質量測值(Class Dependency Quality Measurement; CDQM)：進行基本品質量測之前，必須先分別以線性結合方式將一致性、耦合度、引用限制及被引用限制等四組低階

品質因子加以結合，產生一致性、耦合度、引用限制及被引用限制等四項基層品質量度，再透過公式(1)，將四項基層品質量度結合成高階的類別相依關係品質量測值，結合公式如下所示：

CDQM: Class Dependency Quality Measurement

DCiM: Dependency Consistency Metric

W_1*: Weight of DCiM*

DCoM: Dependency Coupling Metric

W_2*: Weight of DCoM*

DRiM: Dependence Referring Metric

W_3*: Weight of DRiM*

DRdM: Dependence Referred Metric

W_4*: Weight of DRdM*

$$CDQM = W_1 * DCiM + W_2 * DCoM + W_3 * DRiM + W_4 * DRdM$$

$$W_1 + W_2 + W_3 + W_4 = 1 \qquad (1)$$

(2) 類別一般化關係品質量測值(Class Inheritance Quality Measurement; CIQM)：進行一般化關係品質量測之前，必須先分別以線性結合方式將一致性、深度、廣度及覆蓋度等四組低階品質因子加以結合，產生一致性、深度、廣度及覆蓋度等四項基層品質量度，再透過公式(2)，將四項基層品質量度結合成高階的類別一般化關係品質量測值，結合公式如下所示：

CIQM: Class Inheritance Quality Measurement

ICM: Inheritance Consistency Metric

W_{ic}*: Weight of ICM*

IDM: Inheritance Deepness Metric

W_{id}*: Weight of IDM*

IBM: Inheritance Breadth Metric

W_{ib}*: Weight of IBM*

IOM: Inheritance Overwrite Metric

W_{io}*: Weight of IOM*

$$CIQM = W_{ic} * ICM + W_{id} * IDM + W_{ib} * IBM + W_{io} * IOM$$

$$W_{ic} + W_{id} + W_{ib} + W_{io} = 1 \qquad (2)$$

(3) 類別關聯關係品質量測值(Class Association Quality Measurement; CAQM)：進行一般化關係品質量測之前，必須先分別以線性結合方式將一致性、完整性及複雜度等三組低階品質因子加以結合，產生一致性、完整性及複雜度等三組基層品質量度，再透過公式(3)，將三項基層品質量度結合成高階的類別關聯關係品質量測值，結合公式如下所示：

CAQM: Class Association Quality Measurement

ACiM: Association Consistency Metric

W_{ci}*: Weight of ACiM*

ACeM: Association Completeness Metric

W_{ce}*: Weight of ACeM*

ACxM: Association Complexity Metric

W_{cx}*: Weight of ACxM*

$$CAQM = W_{ci} * ACiM + W_{ce} * ACeM + W_{cx} * CxM$$

$$W_{ci} + W_{ce} + W_{cx} = 1 \qquad (3)$$

(4) 類別實現化關係品質量測值(Class Realization Quality Measurement; CRQM)：進行實現化關係品質量測之前，必須先分別以線性結合方式將一致性、引用限制、被引用限制及抽象度等四組低階品質因子加以結合，產生一致性、引用限制、被引用限制及抽象度等四項基層品質量度，再透過公式(4)，將四項基層品質量度結合成高階的類別實現化關係品質量測值，結合公式如下所示：

CRQM: Class Realization Quality Measurement

RCM: Realization Consistency Metric

W_{r1}*: Weight of RCM*

RRiM: Realization Referring Metric

W_{r2}*: Weight of RRiM*

RRdM: Realization Referred Metric

W_{r3}*: Weight of RRdM*

RAM: Realization Abstraction Metric

W_{r4}*: Weight of RAM*

$$CRQM = W_{r1} * RCM + W_{r2} * RRiM + W_{r3} * RRdM + W_{r4} * RAM$$

$$W_{r1} + W_{r2} + W_{r3} + W_{r4} = 1 \qquad (4)$$

(5) 最後透過公式(5)，將相依關係、一般化關係、關聯關係及實現化關係等四項品質量測值結合成類別關係品質量測指標(Quality Indicator of Class Relation; QICR)，結合公式如下所示：

QICR: Quality Indicator of Class Relation

CDQM: Class Dependency Quality Measurement

W_{cd}*: Weight of CDQM*

CIQM: Class Inheritance Quality Measurement

W_{ci}*: Weight of CIQM*

CAQM: Class Association Quality Measurement

W_{ca}*: Weight of CAQM*

CRQN: Class Realization Quality Measurement

W_{cr}*: Weight of CRQM*

$$Q\ ICR = W_{cd} * CDQM + W_{ci} * CIQM + W_{ca} * CAQM + W_{cr} * CRQM$$

$$W_{cd} + W_{ci} + W_{ca} + W_{cr} = 1 \qquad (5)$$

本品質量測模式一共匯集了 15 組基層類別關係之品質因子，且透過線性結合模式產生 15 項基層品質量度及 4 項高階品質量測值，經過三個層次的量度結合作業，最後產生類別關係品質量測指標，稱此量測作業模式為類別關係品質量測*(Class Relationships Quality Measurement; CRQM)*模式，其架構如圖 2 所示。

4.2 品質量測指標衍生的改善作業

類別關係品質量測模式所估算出的類別關係品質量測指標，是協助改善類別關係設計潛在問題與缺失的依據，透過類別關係的低階品質因子、基層品質量度、高階品質特性等三個層次，結合而成的類別關係品質量測值是識別類別關係品質潛在問題與缺失的關鍵。量測值是一種相對性的指標，品質量測值小於 0.5 就可以視為「待改善」的範圍，不過隨著軟體專案對於類別設計品質的要求差異，「待改善」的範圍也可以進行調整，如大型的軟體專案、開發人員眾多或使用單位很重視開發與維護品質等，「待改善」的範圍就應該調整成小於 06 或 0.7。因此，當品質量測指標落在「待改善」的範圍，便可以從品質量測模式中的結合公式，判斷出相關的品質因子，再從量化的品質因子剖析出對映的類別關係之設計項目，進而找出類別關係設計注入的品質問題與缺失，依據問題與缺失可以提出具體的改善措施與監控作業。以下即針對類別關係「潛在問題與缺失」所提出的改善法則：

<Rule 1> IF「類別關係」品質量測指標被判定為「待改善」等級

THEN 進一步分析「相依關係」、「一般化關係」、「關聯關係」及「實現化關係」等量測值是否落在「待改善」的範圍，且透過結合公式(5)，對照找出屬於過低狀態的「類別關係」品質量測值。

<Rule 2> IF「相依關係」品質量測值屬於「待改善」等級

THEN 可以進一步分析出那幾項基層之相依關係品質不良造成「相依關係」品質量測值落在「待改善」的範圍，且透過結合公式(1)，對照找出「相依關係」品質量度值所對應的一致性、耦合度、引用限制及被引用限制等品質因子，再由因子配合找出一致性、耦合度、引用限制及被引用限制等層面潛在的品質問題與缺失，進行修正、改善與監控等措施。

<Rule 3> IF「一般化關係」品質量測值屬於「待改善」等級

THEN 可以進一步分析出那幾項基層之一般化關係品質不良造成「一般化關係」品質量測值落在「待改善」的範圍，且透過結合公式(2)，對照找出「一般化關係」品質量度值所對應的一致性、深度、廣度及覆蓋度等品質因子，再由因子配合找出一致性、深度、廣度及覆蓋度等層面潛在的品質問題與缺失，進行修正、改善與監控等措施。

<Rule 4> IF「關聯關係」品質量測值屬於「待改善」等級

THEN 可以進一步分析出那幾項基層之一般化關係品質不良造成「關聯關係」品質量測值落在「待改善」的範圍，且透過結合公式(3)，對照找出「關聯關係」品質量度值所對應的一致性、完整性及複雜度等品質因子，再由因子配合找出一致性、完整性及複雜度等層面潛在的品質問題與缺失，進行修正、改善與監控等措施。

<Rule 5> IF「實現化關係」品質量測值屬於「待改善」等級

THEN 可以進一步分析出那幾項基層之一般化關係品質不良造成「實現化關係」品質量測值落在「待改善」的範圍，且透過結合公式(4)，對照找出「實現化關係」品質量度值所對應的一致性、實作限制、被實作限制及抽象度等品質因子，再由因子配合找出一致性、實作限制、被實作限制及抽象度等層面潛在的品質問題與缺失，進行修正、改善與監控等措施。

以 CRQM 模式為基礎的持續改善作業流程(如圖 3 所示)，可以標示出類別關係潛在的品質問題與缺失，再配合及時的修訂、改善與監控作業，可以具體的改善類別關係的設計品質，對於軟體專案後續的開發、測試及維護等作業品質應可獲得大幅的提昇，進而有效的強化軟體系統的整體品質。

5. 結論

物件導向設計階段承接物件導向分析所完成的文件，進行軟體系統架構及相關元件組構關係的描繪作業。物件導向設計階段所規劃出的類別元件與類別關係是物件導向設計與程式製作之間的溝通媒介，程式人員、測試人員與維護作業都必須引用類別細部設計與類別關係設計，進行後續的相關作業。由此可知，類別元件與類別關係的重要性，其中類別關係不僅是後續開發、測試與維護作業的依據，對於開發效率與軟體整體品質，更有相當關鍵的影響力。不過，設計不當的類別關係不僅衝擊類別設計品質，也會大幅影響後續的開發與維護作業，完善的類別關係是提昇類別設計品質的必要條件，卻因為缺乏一套具體的改善機制，經常被設計人員所忽略，導致後續開發作業與維護階段極大的困擾。本文中，探究類別關係的關鍵性與衝擊性，且列出類別關係的品質問題與缺失，進而以檢視活動為基礎，提出一套類別關係品質量測 *(Class Relationship Quality Measurement; CRQM)* 模式，以量化的檢視方式，協助識別類別關係的問題與缺失，協助及時矯正問題與改善缺失，具體且有效的提昇類別設計品質。本文以 *CRQM* 模式改善類別關係品質的優勢如下：

(1) 以量測模式產生類別關係品質量測指標。
(2) 以量測指標及量化機制協助識別類別關係設計的品質問題與缺失。
(3) 配合法則式推論，提出品質問題與缺失的改善建議與矯正措施。

本文提出之 *CRQM* 模式具備三項特質：

- 簡化公式—以線性結合公式取代複雜的非線性結合公式。
- 易於標示問題—對照公式，可迅速識別類別關係設計的品質問題與缺失。

- 提高調整彈性—可以隨著關注的重點與應用的差異，彈性調整公式與項目。

誌謝

本論文接受實踐大學 99 學年度「專題研究(蓄積管院研發能量)計畫」(計畫編號：USC-99-05-04004)之補助。

參考文獻

[1] 賴森堂,"改善類別元件模組性之品質量測模式"， 屏東：2003 年資訊技術應用與發展研討會，2003，227-235.

[2] 賴森堂，吳慶陽，蔡閔亘，林俊杰，孫慶文，2007 年 3 月，"類別元件開發輔助工具之規劃與製作"，2007 電子商務與數位生活研討會論文集，2007.

[3] S.W. Ambler, *The Elements of UML 2.0 Style*, Cambridge University Press, Cambridge, UK, 2005

[4] B. Baudry, Y. L. Traon, G. Sunyé, "Testability Analysis of a UML Class Diagram," Eighth IEEE International Symposium on Software Metrics (METRICS'02), 2002 pp.54-66

[5] B.W. Boehm, *Software Engineering Economics,* Prentice-Hall, New Jersey, 1981.

[6] G. Booch, R.A. Maksimchuk, M.W. Engel, B. J. Young, J. C. Kelli A. Houston, *Object-Oriented Analysis and Design with Applications,* Addison-Wesley Professional; 3 edition, 2007.

[7] S.D. Conte, H. E. Dunsmore, and V. Y. Shen, *Software Engineering Metrics and Models*, Benjamin/Cummings, Menlo Park ,1986.

[8] S. Cook and J. Daniels, "Designing Object Systems: Object-Oriented Modeling with Syntropy, Prentice Hall, 1994.

[9] M.S. Deutsch and R.R. Willis, *Software Quality Engineering: A Total Technical and Management Approach,* Prentice-Hall, Inc., NJ, 1988..

[10] D.F. D'Souza and A. C. Wills, *Objects, Components, and Frameworks with UML: The Catalysis Approach,* Addison-Wesley, 1997

[11] R. Fairley, *Software Engineering Concepts,* McGraw-Hill, Inc., 1985.

[12] N.E. Fenton, *Software Metrics - A Rigorous Approach*, Chapman & Hall, 1991.

[13] M. Genero, E. Manso, A. Visaggio, G. Canfora and M. Piattini, "Building measure-based prediction models for UML class diagram maintainability," Empirical Software Engineering, Volume 12, Number 5, pp.517-549, Match 2007.

[14] D. Jackson, **"**Alloy: a lightweight object modelling notation," ACM Transactions on Software Engineering and Methodology (TOSEM),Vol. 11, Issue 2, April 2002.

[15] S-T Lai, "A Quality Measurement Model for Class Component", *proceeding of the 5th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2001),* Florida, USA, 2001.

[16] R.C. Lee and William M. Tepfenhart, *UML & C++: a practical guide to object-oriented development*, Prentice Hall, 2001.

[17] Object Management Group (OMG); Object Constraint Language OMG Available Specification Version 2.0, May 2006

[18] R.S. Pressman, *Software Engineering: A Practitioner's Approach*, McGraw-Hill, New York, 2010.

[19] S.R. Schach, *Object-Oriented and Classical Software Engineering*, McGraw Hill, 2008.

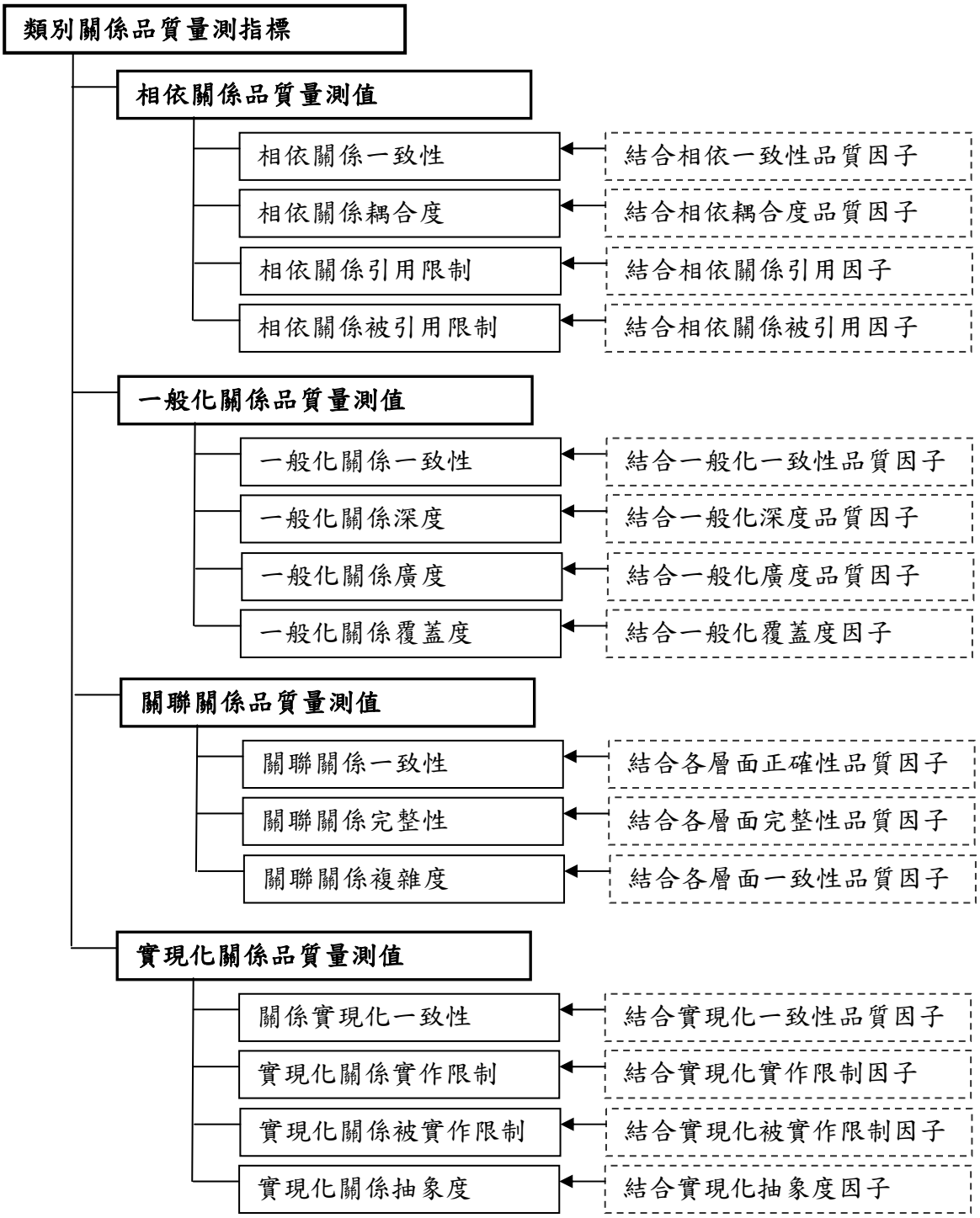

圖 2. 類別關係品質量測模式架構圖

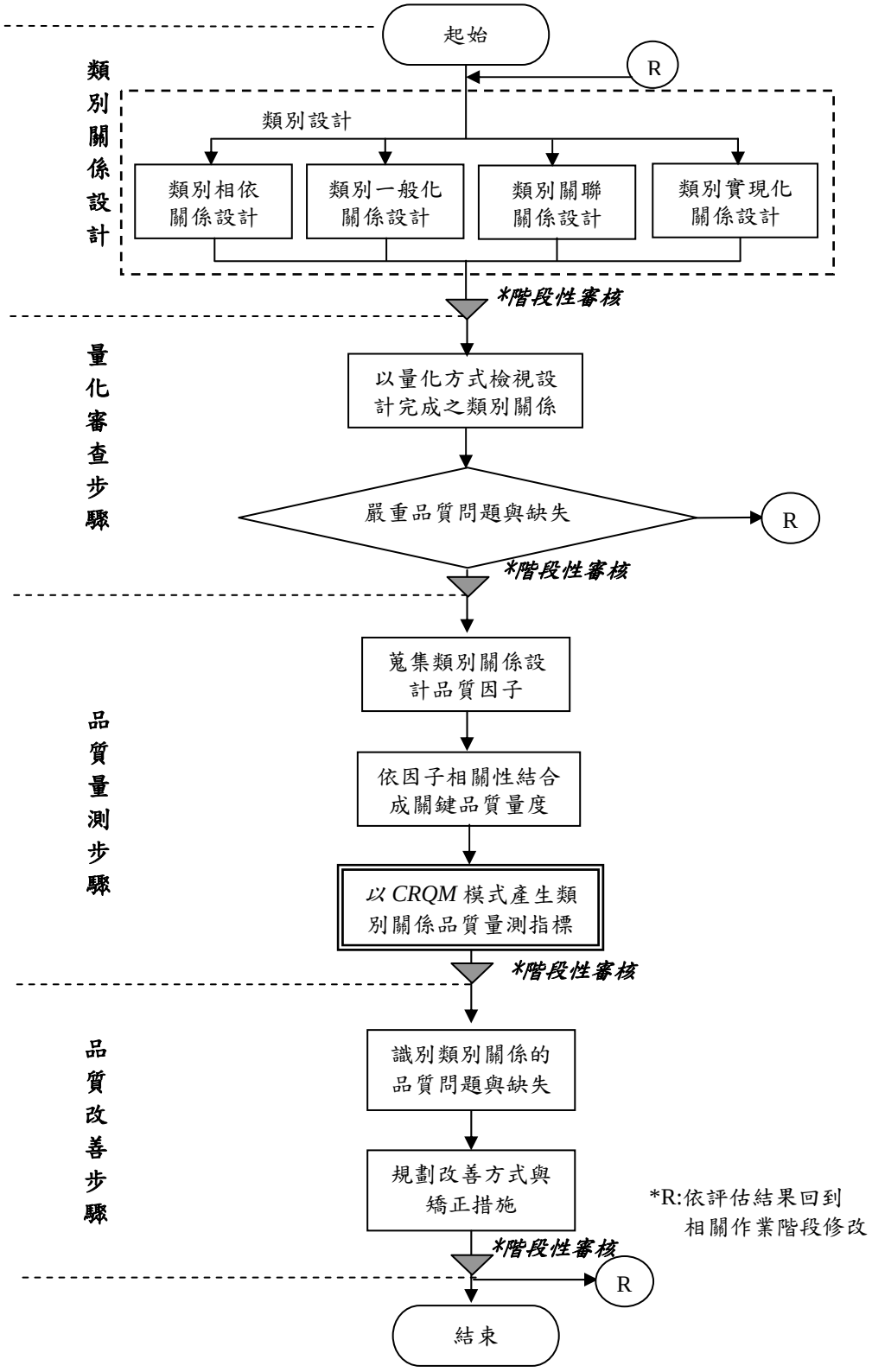

圖 3 以 *CRQM* 模式為基礎的持續改善作業流程圖