The Study of Growth and the Effect of Plant Growth Regulators on Development in Caladium

Yu zhu[1] Hsiu-Fung Chao[2]

1.Department of Horticulture, National Ilan Institute of Technology

2.Tainan District Agricultural Improvement Station, I-chu branch station, Taiwan, ROC.

Abstract

For purpose of establishing the data of characteristics of caladium varieties, twenty-six varieties of *C.bicolor* and five varieties of *C. pictulantum* were used to investigate the tuber weight, leaf number, plant height, leaf shape and the speed of emergence rate. From the data we collected, there was no obvious correlation in tuber weight, leaf numbers and plant height between same or different varieties. Heavier tubers were not necessarily had more leaf numbers or owned higher plant height. BA treatment increased the leaf numbers, but not as much as terminal bud removal. GA_3 treatment could raise the flowering rates and flowering numbers as well as induce the leaf shape into narrow.

Key Words: Caladium, mean leaf numbers, mean petiole length, emergence percentage,

BA, GA_3,flowering percentage, flowering numbers

A strategy for integrating software testing and development process

Ming-Yieu Kuo
Computer Center, National I-Lan Institute of Technology

ABSTRACT

How to incorporate software testing into each phase of the development process efficiently is a critical factor for improving software quality. Current related works are all integrating methods either based on the technological aspects or the artifacts structure aspects of the process. Nevertheless, from the standpoint of management, they are lack of a strategy to coordinate the conflict in resources between the development and quality teams.

This study starts with the goal of total quality management, and aims at guiding development and testing with the help of Demin's PDCA quality cycle principle. Moreover we propose a framework, based on XML, to initiate the documentation structure of development and testing, and, furthermore, with the aid of hyperlink to achieve traceability and monitoring purpose.

Keywords: software development process, software testing, software quality, XML

一、前言

在 Internet 大環境趨動下，資訊科技應用已產生莫大衝擊；對一般使用者而言，週期性、定點、定性(報表或螢幕查詢)的資訊來源已躍昇為即時、行動、多元性(含多媒體)的網路新紀元；對軟體開發者而言，更是面臨 web 應用多變性且要求縮短開發時程、技術標準的推陳出新、以及多樣化的開發工具等等新挑戰。

目前從事軟體開發的專業人員，負責分析規劃較具工作經驗者，習慣於傳統直線式開發流程，而近年來新進人員一般從事程式撰寫工作者，雖已普遍熟悉物件思維，可跳脫傳統模式，卻也囿於學校正規教育偏重於個人的設計技巧，長期忽略整體團隊軟體工程的開發訓練，新舊合作益形困難，祇能邊做邊修正，遑論建立一個可持續改善軟體品質的開發環境。

依據軟體工程領域知名大師 Pressman 在[1]中所提到：軟體工程的一切作為都是植基於有組織地實踐品質目標，工具、方法與過程三者層次性架構整個軟體工程。檢視當今軟體開發工具與方法不外乎是視覺化，元件化的物件導向整合開發環境已普遍被接受，對於個別設計工作已無大礙，然而過程卻是連接技術層面的接合劑，用來建構軟體專案管理，有效應用技術方法，產生工作產品是攸關整個品質良瓢的關鍵所在。尤其正當軟體開發過程由傳統直線式進化成為反覆漸增(iterative and incremental)式之際，普遍存在著新舊思維混雜所帶來的困擾；再者，軟體測試是軟體品質保證的最後一道防線，在傳統直線式之下，開發活動節次分明，測試階段安排在程式設計之後，然而在反覆漸增的發展模式需求不易界定，連帶地測試工作也分散在每一回開發循環，處在這類發展環境，尤其須要將測試活動適時地緊密結合於系統開發過程[2]始能發掘軟體缺失。

本篇論文主要目的在於，從提昇軟體品質的思考角度出發，探討軟體發展過程與測試環境的整合之道。深入瞭解在反覆漸增的開發過程如何融入測試活動，希望藉助於 web 的技術標準，規劃整個活動的資訊傳遞機制，以資訊內容為重心，設計軟體產出物(artifacts)提昇軟體品質。一者有助於開發人員的協調合作，進而架構一套有助於協調合作並提高測試效能的軟體開發環境。本文下一段落是探討相關研究與可行技術，經過評估後，於第三段落提出建構方法與環境架構，最後並說明結論與未來研究方向。

二、相關研究

軟體工程的終極目標在於生產力和品質的持續提昇。而有效的軟體品質活動應該做到[3]：

。融入於軟體產品生命週期的每一階段

。符合軟體工業標準

。活動可依規劃內容以書面文件結構化表達

。可重覆使用並且自動化

。可以量測

因此在探討如何整合軟體開發過程與測試環境之道，首先就是要確立整合架構是以品質為中心，建立一套可持續改善軟體品質的生產環境。

(一)、軟體開發過程之整合方法

分析上列要件，軟體開發的每一過程，都要透過健全的文件記錄機制提供管理活動所需的資訊，並且可以支援不同階段其他開發成員參考再運用，以達到品質量化的目標。

品質管理大師戴明曾指出: 大多數的問題是由於作業過程(process)所引起[4]，目前軟體工程普遍流行物件導向思維，以元件為基礎(component-base)的設計架構，採用快速應用程式設計(RAD：Rapid Application Design)方法，以反覆與漸增流程開發軟體系統，尤其隨著統一化塑模語言(UML：Unified Modeling Language)已經成為工業標準之後，開發過程中分析與設計階段都能維持一套共通模式語言，且能視覺化表達設計結果，誠然大大提昇溝通效果。然而開發過程的管理層面，例如測試驗證、組態管理、時程控制等等卻始終缺乏一套標準，究其根由，主要在於開發工具之間尚未能有效互通交換資訊，造成開發過程無法上下游整合。

有關軟體工程環境的整合方法，相關的研究有：[5]文提出資料整合形式提供工具之間的資料共享，建立專案資料儲存庫與可供每個開發成員共通使用的程序訊息伺服器，利用 FrameMaker 文書處理軟體之 API 開發共同編輯器，可圖文處理。提出高品質的語意架構可整合不同形態的軟體產出物。[6]一文從超文本(Hypertext)觀念及軟體文件的存取便利性來研究如何讓開發人員可快速地進入專業狀況，特別的是以 3D 視覺化資訊表達專案資訊，並利用可擴充標示語言(XML：eXtensible Markup Language)為基礎來架構開發環境，祇維護以上層資料(metadata)形式表達的產出物之位置及存取資訊。軟體產出物可以放在原始產生位置，例如原始程式碼放在建構管理系統，設計文件放在文件管理系統；亦即 CHIME 並不處理產出物的儲存問題，設計成員可繼續使用既有工具來存取 CHIME 所提供開發資料及鏈結超文本的機能。

開發過程的整合方法除了上述兩文的資料整合之外，[7]的研究方向在於處理過程的方法指引，制定整體的塑模過程、設定程序及執行等三個指引範圍，以 C++語言開發實作基礎架構(implementation framework)完成工具與處理模型的整合並建立作業引擎的實作架構及系統改變的整體支援。儘管試作環境尚以小型系統為主，使用者反應表示在開發過程都能接受引導，順利運用整合工具。

(二)、處理軟體開發產出物的新典範

軟體開發的活動內容，主要可分為以處理過程為導向(process-oriented)和產品為導向(product-oriented)等兩種，前者以管理協調為主，後者重點在於設計製作過程的既定程序，兩者具有互補關係[7]，而依據二之(一)節的研究發現，串接上述兩種活動的關鍵要素就在於產出物的共用機制。

解決產出物的共用機制，傳統上都以儲存庫(repository)研究為主，近年來是以產出物內涵為研究主題。首先是從文件再利用為切入角度，主張運用 XML 設計軟體開發產出物[8]，指出文件的內容及結構適合用 XML 分離，並舉例說明如何運用 XML 建立美國國防部所制定的軟體文件標準(MIL-STD-498)、軟體組態檔(configuration file)、軟體編譯的 Make file 等等，同時展示將 Java 的原始程式碼加上標示(tag)，除了自動產生說明文件外並可適用於 web 環境，利用這些 tag 可進一步轉換成其他特定的程式碼文件。[9]文探討在 Internet 平台分派軟體設計工作所遭遇的問題，特別是工具之間交互運作所需的交換資料，針對 UML 模型以 XML 語言建立交換格式，(UXF：UML eXtensible Format)作為 UML 之結構性資料編碼與交換的機制。儘管由 EIA 及 ISO 已共同定義了 CSIF(CASE Data Interchange Format)用於 CASE 工具間軟體塑模資訊交換，但 UXF 的最大優點在於它是直接自 web 環境延伸而來，可容易被已熟悉 HTML(HyperText Markup Language)或 SGML(Standard Generalized Markup Language)的人接受，適合分散式模型管理。探討 Internet 上文件的組成與互動機制的[10]文，提出 active document 的解決方案，有別於目前常用的二進位格式交換(如微軟的 COM 元件)，即在 active 文件上不含特定執行碼，取而代之的是以 XML 表達文件內容及 active 部份，可動態地載入可執行模組，以 Java class 動態地與 XML 文件上元素互相結合，提供該元素的瀏覽顯現(rendering)行為。它是透過自行開發的 XMLC(XML Compiler)將 XML 文件編譯成樹狀 DOM(Document Object Model)模組，再和特定應用系統所設計的 XSL(eXtensble Stylesheet Language)排版樣本，經 XSL 處理器解讀後，產生 Java 物件後可供執行，因此可依應用環境彈性地定義排版樣本，尤其軟體工程使用的特定符號(如 UML 的各式圖塊)可由抽象語法轉換成行為語意，做到文件分析與轉換目的。

(三)、軟體測試之探討

軟體測試是為了發現程式錯誤而執行程式的過程，所謂好的測試案例(test case)是指具有較高的發現未暴露錯誤機率者，而測試過程所收集的資料可提供軟體可信度與整體的軟體品質的指標。必須在開始運作之前就要有妥善的計劃，但是徹底完全的測試目標是不可能的，因此如何在有限的人力物力資源下於一定的

時間內完成最大測試涵蓋範圍，就成為軟體開發的一大挑戰[11]。

測試案例設計方法可分為白箱測試與黑箱測試，前者使用程式的控制結構來產生測試案例，後者集中於軟體的功能需求之驗證。若就軟體開發階段所進行的不同測試工作則可分為單元測試、整合測試、系統測試及驗收測試。處於軟體複雜度與日俱增而開發時程卻要求縮短的壓力下，如何協助測試人員有效運用測試工具已成為最重要課題。

在[12]中透過專家與學者的 workshop 討論內容，可瞭解自動化測試的耗費高昂且找出的錯誤大約是 6%至 30%，其他大部份仍由人工測試找出，自動化測試適合在執行初期測試，但是到了後面階段還是應由有經驗技巧的人員進行較妥。同時提到考慮測試案例的維護問題時，一般常用的 capture/replay 工具不能用來產生測試案例，因為一旦修改所設計軟體的使用者圖形介面，原先的測試案例就無法使用了。為了跳脫 capture/replay 工具產生固定測試案例的困擾，[13]之研究就提出有效分離測試腳本(test script)實作與設計的構想，希望將測試案例可以更高的抽象化表達，建議使用電子試算表製作測試案例，經由處理程式解釋後，由 capture/replay 工具執行該項測試及驗證結果。由於每個工具所用的腳本語言都不盡相同[13]的構想就不易實現。尤其攸關可用性評估更是須要可反應使用背景(context)的測試語意[14]，而以目前語意轉換的機制而言，當然非 XML 語言莫屬了，[15]一文就是採用 XML 語法來表達測試案例，它只設計 do what 而不是如何執行測試，依據不同平台的每一工具腳本設計轉換規則，利用 XML 可自訂 tag 來表達受測領域的專用詞，定義測試工作及檢測準則，並描述測試案例，最後再由轉換規則產生執行測試工具的腳本，進行自動化測試。

三、以軟體品質為主導的整合架構

從上一章節的相關研究可瞭解：透過軟體開發過程產出物的共用機制，尤其是以 XML 語言建立儲存結構，的確有助於整合開發過程與測試環境。然而實務工作畢竟有別於研究架構，現今軟體開發過程普遍以反覆完成多回軟體成品(builds)的模式，在相關研究中就沒有考慮；反之，微軟公司[16]的開發模式就是在專案進行過程中反覆地引導開發工作同步進行，並週期性漸進地加強產品穩定性。由此可見測試工作為發揮提高軟體品質的應有功能，首先就要配合其他開發階段的進度，否則在生產為重的觀念下常犧牲測試計劃忽略品質。如何兼顧生產與品質的議題正是本文探討如何整合開發過程與測試活動的主要方向。

(一)、整體性軟體開發架構

為了強化軟體開發過程與測試活動之間的緊密結合，發揮相乘效果，根據以上研究結果，吾人提出建置整體軟體開發架構的組織機能並圖示架構圖[圖 1]。

其組織機能如下:

1、web 化協同合作開發環境

利用 Internet HTTP 協定，統合各開發階段所有成員之間的訊息傳遞，可就 web 應用程式之開發工具，設計專案管理訊息之傳遞機制，專案人員可以瀏覽器操作進度時程跟催，達到同步工程目標。

2、開發程序以品質管理為導向

在反覆與漸增開發模式下，以 PDCA 為指導原則，主導各回循環的生產活動，所有開發程序的細步設計皆以品質循環原理為決策依據，整合開發過程與測試活動。

3、產出物資訊背景化與再利用

運用 XML 語言可自由訂定上層資料並動態載入執行模組功能，可將測試程序設計"使用背景化"(contextualize)，跳脫傳統以抽象命令表達測試腳本，尤其可利用 XML 文件轉換及超文本的機制串接各式關聯文件，充分再利用產出物資訊。

4、專案決策的因果模型

開發過程除了收集統計之量化資料，更可利用 XML 超鏈結功能追蹤問題所含蓋的使用個案與測試個案，進而可設計軟體量度之因果模型，提昇專案決策能力。

(二)、整合性軟體開發環境之建構理念

上節(一)所提之開發架構及組織機能，揭露出建構整合性軟體開發環境的指導理念，一旦完成建置此開發環境，其特色與優點在於：可依據 PDCA 原理擬訂軟體開發與品質活動的細步實施步驟，並且利用 XML 的「詮釋語言」(meta-language)特性，主導產出物的管理架構，可讓品質活動完全融合於開發過程的每一階段，達到整合目標。比較第二節的相關研究與本文所提架構的類似之處在於 XML 的軟體工程上應用，然而本文獨到特色是結合 PDCA 原理指導軟體開發與品質活動，亦即在設計產出物資訊的 XML 架構時，融入 PDCA 的管理因子，以求達到全面品質的目標。

1、軟體開發的全面品質管理

為了在有限的資源條件下於壓縮的時程內發表產品，軟體開發團隊常常棄守既定的品質政策，遷就於滿足市場或使用者的要求，如此一來造成日後多次的修改版本。因此如何讓設計人員與測試小組能夠齊一步調，當面臨時間壓力開發過程必須有所縮減的抉擇時，雙方可獲致共識且能達到應有品質水準，實有賴於一套完善的品質管理政策為依歸。探究軟體品質保證活動的主要工作可分為：軟體測試、品質控制與軟體行態管理等三大項目[2]，而依據[17]研究，對於軟體需求

規格及程式的查核，亦即品質控制工作是與測試同等重要，而行態管理的重點在於變更控制與版本維護。因此整體的品質活動實則涵蓋系統開發的每一階段每一成員，從單元測試、整合測試、系統測試到靜態程式碼、需求規格的人工查核，以及串接整個軟體生命週期的更改維護，也正是戴明[18]全面品質管理的具體實現。[19]文中分析反覆式開發過程的品質管理活動，於四個開發階段的活動內容，可進一步用戴明的品質循環原理 PDCA：Plan(規劃)、Do(實施)、Check(查核)、Act(持續改善)擬訂系統開發與品質活動整合的細部實施步驟，例如在建構階段，進入程式碼設計階段，一旦類別方法設計後，就可設計單元與整合測試腳本(Do 循環)，然後漸增地完成次系統整合，執行軟體成品(build)的測試(check 循環)，在移轉階段，由使用者執行驗收，將發現問題並排定優先順序與修復計劃(Act 循環)，所以可就 PDCA 原理主導整合的架構。

2、以 XML 為主導的產出物管理架構

開發過程的產出物其資料格式有文字、符號、圖形等。其結構性有例行表單的結構化，也有敘述性的半結構與非結構化，且由於許多設計不祇是技術方面的記錄更可用於專案管理的追蹤考查。因此資料的再使用性與多向展現就十分普遍。根據本文二之(二)節之探討，XML 語言於軟體工程的應用，目前是用於技術資料的內容與結構之設計，重點放在文件內涵與資訊交換。惟產出物的管理架構依[19]文指出其機能應有：支援活動追蹤、量化統計基礎及資訊傳遞平台等，技術性與管理性兩者兼備，再加上以 XML 為主導設計產出物的儲存結構，可充分發揮調合功用，讓品質活動可完全融合於開發過程的每一階段。

現就產出物管理機能的角度探討如何運用 XML 語言：

開發活動的追溯情形存在於測試小組和設計成員之間，追蹤程式缺失與問題所含蓋之使用個案(use case)，因此測試個案與使用個案，都要以電子媒體儲存，再以 XML 之文件格式定義(DTD：Data Type Definition)文件架構，進而可依報表或查詢格式以 XSL(eXtensible Stylesheet Language)處理器依照排版樣本，將資料重新排列與轉換成所要格式並加以展示，形成測試個案與使用個案之間相互參考的追蹤矩陣效果。而測試個案可參考[15]作法，利用 XML 的標示來表達受測領域的專用語彙，定義檢測準則。為了提供決策所需資訊及確認軟體可靠度，測試後產生的缺失及問題報告，亦須記錄與控管，尤其後續的迴歸測試(regression test)更需要一套完善的測試資料收集與回復程序，利用 XML 的超鏈結：Xlink 及 XPointer 分別連結文件內部與外部文件，且一次可鏈結多個對象進而是跨文件且指定段落[20]的超鏈結法，此一功能可用來統計錯誤數量與追蹤問題的關聯性，因此除了收集傳統的軟體量度(metrics)做到以統計模型預估軟體品質與成本之外，也有可能提昇為以因果模型為主導的產出物管理架構軟體工程決策模式

[21]。以 XML 上層資料(metadata)表達產出物資訊，適合在 web 環境跨平台傳送開發過程的資料，可超越傳統特定資料庫的束縛，透過 web 的傳輸協定，就可建構開發團隊的協同合作環境。

(三)、整合性軟體開發環境之系統規劃

對於軟體開發人員與品質管理小組而言，其工作重心在於有效運用開發或測試工具，至於有關開發環境中使用之開發文件的 XML 格式製作，以及整個環境的產出物訊息傳遞機制，應不是他們的技術重點，因此吾人實有必要先行規劃整合性軟體開發環境，讓每個開發專案都能在系統化的流程引導下，發揮全面品質管理的精神，提昇生產力提高軟體品質。

系統方塊圖如[圖 2]所示，各子系統之功能分述如下：

1、文件製作子系統

文件製作子系統主要是基於 XML 語言規範，對軟體文件產出物的 XML 架構、內容、樣式等三個要素進行管理，主要包括三個模組。

(1) 文件結構編輯模組

以圖形化的使用者介面(GUI)，提供專案管理人員定義文件產出物的結構，並可自動轉換成文件格式定義(DTD)或 XML 綱要(schema)，可作為驗證 XML 文件的完構性及有效性。

(2) 文件內容編輯模組

具有一般文書處理軟體的編輯功能，並應考慮統合特性，例如在微軟 Office 架構之下，可互通 Word、Excel 的檔案內容，並可趨動 Internet 元件充分結合 Web 應用程式。

(3) 文件樣式編輯模組

配合文件產出物在不同開發及測試階段，對不同人員對象的展示需要，設計不同樣式編排，亦即透過 CSS 或 XSL 方式來顯示。

2、文件萃取子系統

文件萃取子系統的主要任務是對 XML 文件結構及內容進行加值處理，可以透過「文件物件模型」(Document Object Model，DOM)設計 XML 的應用程式，適用於軟體文件存取處理，主要包括二個模組。

(1) 剖析檢測模組

利用已定義好的 DTD 或 Schema 檢測軟體文件在語意或結構上的完構性及有效性，目前以微軟所提供之 MSXML 剖析器(Parser)，可進一步以 API 或 ActiveX 方式開發應用程式。

(2) 轉換檢索模組

轉換檢索模組是將文件內容，依據既定的結構轉成現有資料庫方式儲存，

如此可運用既有資料庫管理系統對文件進行存取、查詢、管理等處理，以微軟新版的 ActiveX Data Object (ADO)模組，可以統合既有 Access 或 SQL Server 與 XML 文件，因此可以 SQL 語法檢索文件內容。

3、品質保證(SQA)子系統

SQA 子系統的目標是希望軟體發展過程可持續改進，專案開發經驗可文件化記錄下來，並且是參考 PDCA 的指導原則規範整體開發過程，主要包括兩個模組。

(1) PDCA 指導模組

可就專案開發規模大小分別擬訂檢核表(Check List)，檢視在 PDCA 原理的應用架構下，軟體測試與開發過程的整合步驟，從訂定軟體品質政策開始，經過需求設計與系統建置，使用靜態檢驗或動態測試方法，最後進入問題矯正及預防措施，並對系統可用性的回饋意見做適當處理，完成 PDCA 管理循環，達到持續改善軟體品質的目標，建立檢核表可提供專案管理者當面臨測試與開發行程同步的衝突問題時，可作為決策的參考準則。

(2) 軟體量度模組

本模組在於提供客觀的量測數值，用來評判軟體開發過程、專案執行以及軟體成品本身的品質程度，模組建立程序通常可依下述三個步驟：

步驟一、建立一套量度基準(metrics baseline)

依據組織特性，選定品質要求項目的重要性優先順序，擬訂量測的目標以及量測結果的報告內容，進而確立所要收集的量度資料，並規劃量測程序，例如量測每千行程式碼的缺失率，或採用功能點為計算基準。

步驟二、實施量測活動(measure)

計算量測結果產生量度(metric)

步驟三、記錄與評估，改善過程

隨著專案進行過程，不同階段產生之成品(builds)，可記錄量度進而可產生統計指標(indicators)，據以評估生產力與品質程度，最終任務在於可持續改善開發過程。

品質量度應儘可能自動化產生，然而仍無法免除依賴人工主觀計數與決判，此一現象尚出現於，本模組提供數量化參考資料予 PDCA 指導模組比較時之半自動化應用。

4、專案管理子系統

本子系統是整合開發環境的樞紐，可存取 XML 文件，並萃取管理資訊，且能結合 PDCA 檢視表監測專案開發進程，同時在控管過程記錄品質量度資料，主要含蓋兩個模組。

(1) 開發過程控管模組

本模組主要是進行專案之時程與成本控管，在反覆漸增的發展模式下，以

使用個案為主導，將專案開發工作分解成樹狀工作清單(Work Breakdown Structure)，工作項目且可歸屬於每次循環產生的成品(builds)，如此可利於時程與成本之控管。

(2) 測試過程控管模組

一旦完成使用個案的初步歸範設計，就可著手擬訂測試計畫，設計測試個案，並且可追蹤所含蓋的使用個案，透過資料庫儲存測試報告與問題報告，如此可利於測試過程管理。

上述兩個模組的活動項目皆有記錄所屬 PDCA 的工作範疇，可隨時對照 PDCA 檢視表，並可依據所要量測的品質因子記錄儲存其相關量測值達成 SQA 目標。

5、WEB 介面子系統

本子系統是設計 WEB 應用程式，提供軟體開發人員與品質保證小組，可利用 Internet 平台與專案管理者溝通連繫，可例行性登錄及查詢開發工作的進程報告，而管理人員亦能透過 E-Mail 進行跟催，希望可建構一個協同合作的開發環境。

四、結論與未來研究方向

目前軟體開發模式是以元件化、反覆漸增為主流，在漸增過程中，需求不易掌握，連帶地測試工作也分散在每一回開發循環，造成開發人員與品質小組之間的協同合作益形困難，因此如何整合開發過程與測試程序，就成為專案的成功關鍵因素。

探討相關研究文獻，發現都是單就過程的技術層面或開發產出物的結構面，分別尋求整合方法，然而缺乏一套以管理角度來調合開發人員與品質小組之間的資源分配衝突問題。

本文以全面品質管理之目標為出發點透過戴明 PDCA 品質循環原理，指導開發設計與測試活動，並提出以 XML 為基礎架構，主導開發與測試資訊的文件組織，進而運用超連結功能，達到追蹤控管的管理目標。軟體開發環境中已普遍採用之視覺化塑模工具，以統一化塑模語言(UML)為工業標準，目前以 IBM 為主導聯合許多軟體大廠，推動所謂開放應用系統交換標準(XMI Opens Application Interchange)，是以 XML DTD 訂定 UML 的設計元件，達到開發工具之間資訊交換，因此未來軟體開發工作，上層的分析設計所採用之工具將是以 XML 為規格的主導。而下層的測試與驗證，若要充分發揮效能，確實有必要依本文所主張的 XML 基礎架構作整體規劃。未來研究方向：近程將整合軟體開發環境的 XML 基礎架構，以一般化開發過程及測試活動為建置內涵，中遠程研究目標就是整合

開發工具的 XMI 標準。

參考文獻

1、Roger S. Pressman（1997）, Software Engineering: A Practitioner's Approach, 4Th Edition, pp.22-53, McGRAW-HILL.

2、William E. Lewis(2000),Software Testing and Continuous Quality Improvement, pp.89-102, AUERBACH.

3、Lawrence E. Niech,(1998),Practical Approach of Software Quality Assurance to Commercial Software, HandBook of Software Quality Assurance, 3rd,Edition, pp.513-553, Prentice Hill.

4、林公孚(2000),「確保品質要從過程開始」, 品質管制月刊, 第 36 卷,第七期, 第 79-80 頁.

5、Steven P. Reiss (1999), "The Desert environment,"ACM Transactions on Software Engineering and Methodology, Vol. 6, No.4, pp.297-342.

6、P. Devanbu, Y. - F. Chen, E. Gansner, H. Muller and J. Martin, "CHIME: customizable hyperlink insertion and maintenance engine for Software engineering environments," Proceedings of the 1999 international conference on engineering, May 1999, Los Angeles, CA USA, pp.473-482.

7、Klaus Pohl, Klaus Weidenhaupt, Ralf Domges, Peter Haumer ,Matthias Jarke, Ralf Hamma(1999), "PRIME--toward process-integrated modeling environments," ACM Transactions on Software Engineering and Methodology, Vol.8, No. 4, pp.343-410.

8、Emesto Guerrieri, "Software Document Reuse with XML," Proceedings of 5Th International Conference on Software Reuse, Jun 1998, pp.246-254.

9、Junichi Suzuki, Yoshikazu Yamamoto, "Managing the Software design documents with XML," Proceedings of the sixteenth annual international conference on Computer documentation, Sep 1998, Quebec Canada, pp.127-136.

10、Luca Bompani, Paolo Ciancarini and Fabio Vitali, "Software engineering and the Internet: a roadmap," Proceedings of the 22nd International Conference on The future of Software engineering 2000, Jun 2000, Limerick Ireland, pp.303-315.

11、Boris Beizer(1996), Software Testing and Quality Assurance, pp.12-15, International Thomas Computer Press.

12、Cen Kaner, "Improving the Maintainability of Automated Test Suites,"

Proceedings of Quality Week’97.

13、Edward Kit, “Integrated, Effective Test Design and Automation,”：http：//www.sdmagazine.com/breakrm/feature/s992f2.shtml.

14、郭明裕(2000), “電子商務軟體可用性的測試與評估” 樹德科技大學校園無線電子商務研討會論文集，pp.25-30.

15、Chang Liu, “Platform-independent and tool-neutral test description for automated software testing,” Proceedings of the 22nd international conference on software engineering, Jun 2000, Limerick Ireland, pp.713-715.

16、Michael A. Cusumano and Richard W. Selby, “How Microsoft builds software,” Communication of the ACM, Vol 40, No6. 1997, pp.53-61.

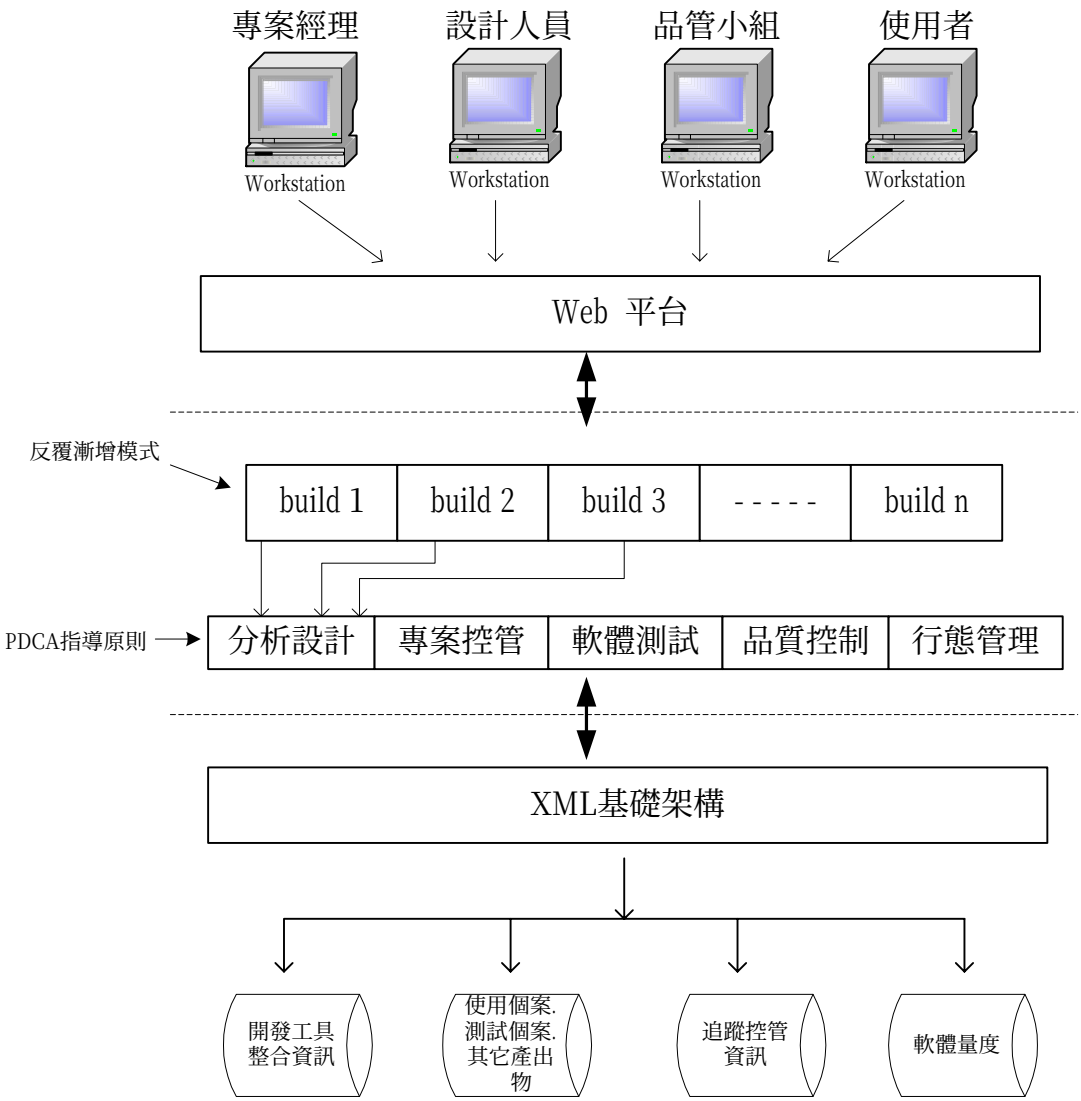
17、Mary Jean Harrold, “Testing：a roadmap, ” Proceedings of the 22nd international conference on software engineering, Jun 2000, Limerick Ireland, pp.61-72.

18、Deming, W.E(1986). “Out of the crisis,” Melbourne Sydney：Cambridge University Press.

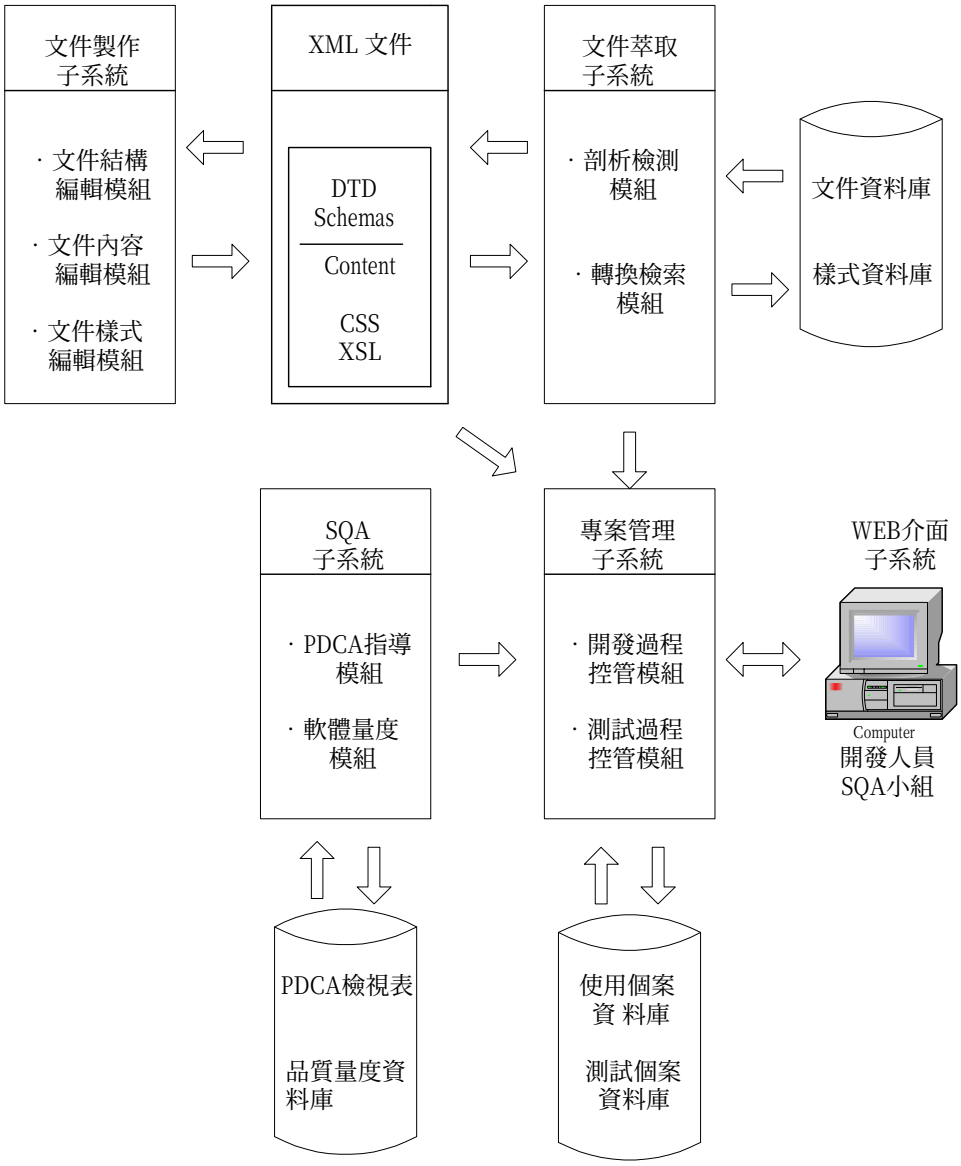
19、郭明裕(2000), “在反覆式發展過程提高軟體品質之研究, ” 中華民國品質學會第 36 屆年會論文集，pp.487-491.

20、Kurt Cagle(2000), XML Developer’s Handbook., SYBEX.

21、Norman E.Fenton, Martin Neil, “Software metrics：roadmap,” Proceedings of the 22nd international conference on software engineering, Jun 2000, Limerick Ireland,. pp.357-370.



[圖1] 整體性軟體開發架構



[圖2]　整合開發環境系統方塊圖