軟體工程標準實際應用效用之分析

Techniques for Evaluating the Effectiveness of Software Engineering Standards

王有利 Yu-Li Wang

元智大學資訊工程系

中壢遠東路 135 **號**

Computer Engineering & Science Dept.

Yaun-Ze University, Taiwan

范金鳳 Chin-Feng Fan

元智大學資訊工程系

中壢遠東路 135 **號**

Computer Engineering & Science Dept.
Yaun-Ze University, Taiwan

E-mail: csfanc@saturn.yzu.edu.tw

摘 要

在軟體發展的過程當中，如何在眾多的標準中選擇適合、有效的標準是有效的是一重要議題。本文提供了一套評估標準實際應用效益的方法，結合靜態與動態的分析。我們的分析方法分為形式(Syntactic)**、內容**(Semantic)**及實際效用**(Pragmatic)**三個層次。靜態部份提供形式及內容的分析，動態方向則提供實際應用效用上的分析。靜態我們利用** XML **超文件的優點，重新建立標準的結構、進行分析。動態上則採用軟體生命週期程序模擬。同時，我們以** UML **建構出一具通用性與可重複使用性的軟體程序模擬框架。藉由靜態分析所辨識出標準間的差異性或標準內的主要議題加以模擬以瞭解其實際影響的效果。我們以** IEEE1012-1986 **年版與** IEEE1012-1998 **年版的比較為一個應用案例以說明本方法之有效性。**

關鍵字：軟體程序模擬、UML**、物件導向框架**

Abstract

How to choose an appropriate and effective standard from so many existing software industrial standards is an important issue. In this research, we provided a systematic approach to evaluate the effectiveness of software engineering standards. In our method, we combined static and dynamic analyses, and included syntactic, semantic, and pragmatic levels of analyses. In static analysis, we utilized XML markup to tag the structure of the standards, and followed a set of criteria to analyze standards; thus, the effectiveness of the syntactic and semantic aspects of standards can be investigated. For dynamic analysis, we developed a software life cycle process simulator to reveal the pragmatic effects of standards. We first designed a generic and reusable process simulation framework in UML. Using it, we can quickly simulate different software processes by plugging in different formulas and data to observe the effects of the major differences or issues identified from the above static analysis. A case study comparing IEEE1012-1986 with IEEE1012-1998 version was also presented to demonstrate the effectiveness of our proposed approach.

Keywords: software process simulation, UML, object-oriented framework.

一、緒論

90 **年代軟體工程標準就已遠超過** 250 **種**[9]**，如何在眾多的標準中選擇適合的標準，又如何確定所選擇的標準是有效的是一重要議題。本文發展出一套有系統的整合比較分析技術。我們的分析方法分為形式**(Syntactic)**、內容**(Semantic)**及實際效用**(Pragmatic)**三個層次。靜態部份提供形式及內容的分析，動態方向則提供實際應用效用上的分析。靜態上著重分析比較標準的邏輯架構特性，如：可遵循度或是明確性等性質。動態上則藉由軟體程序模擬靜態分析的結果及議題可能產生的績效狀況。我們同時設計了一套具通用性且可重複使用性的軟體程序模擬的物件導向框架**(framework)**，使用此框架可快速地建構不同的標準的程序模擬。我們以** IEEE1012 **新舊版間的差異比較為本研究的探討案例。**

二、相關研究

過去標準分析比較研究也有許多，例如 Fenton[2]，Herrmann[3]，Sommerville [10]等學者都曾提出過多種標準的分析評估方式。這些學者所提出的方式多採用等級化的方式分析比較所要比較分析的標準。

物件導向框架(framework)為針對相似的問題所提出的通用設計方式，包含抽象或具體的物件類別集合[5]，具有可重複使用的特性。在框架(framework)研究方面，Young Jong Yang 等學者[11]提出以 UML 為基礎的物件導向式的框架(framework)發展程序，包含四個軟體發展階段: 分析、設計、實作與測試。分析階段針對一組相似的應用需求規格擷取共同的功能來建立框架。在設計階段中包括定義在框架中的類別與熱點(hot spots)，並將框架以文件表現出來。我們框架的建構方法則是改良其提出的方法中的分析與設計階段的程序。

軟體程序模擬相關的研究眾多，例如 Kusumoto [7]的 Petri Net 似模型，Chin Lin[8]在 SPL 發展的 SEPS 具回饋特性。本研究採用數項 Lin 所提的模擬公式。

三、研究步驟

我們的研究步驟如圖一所示。

步驟一至三：

分析標準並建構標準標記結構，可建立對照表比較不同的標準。此步驟有助於以下靜態的分析和動態的模擬。此為建立形式層(syntactic level)分析。

步驟四：

靜態（定性）分析：我們在內容或語意層(semantic level)分析上，將評估標準的明確性與可遵循程度。在可遵循度方面，我們分析在標準的要求與準則對發展者與審查者來說是否清楚。明確性方面，我們則是將需求分類，分為程序、產品與資源三大類，分析發展者與審查者是否能很容易的瞭解標準中所要求的各項需求。

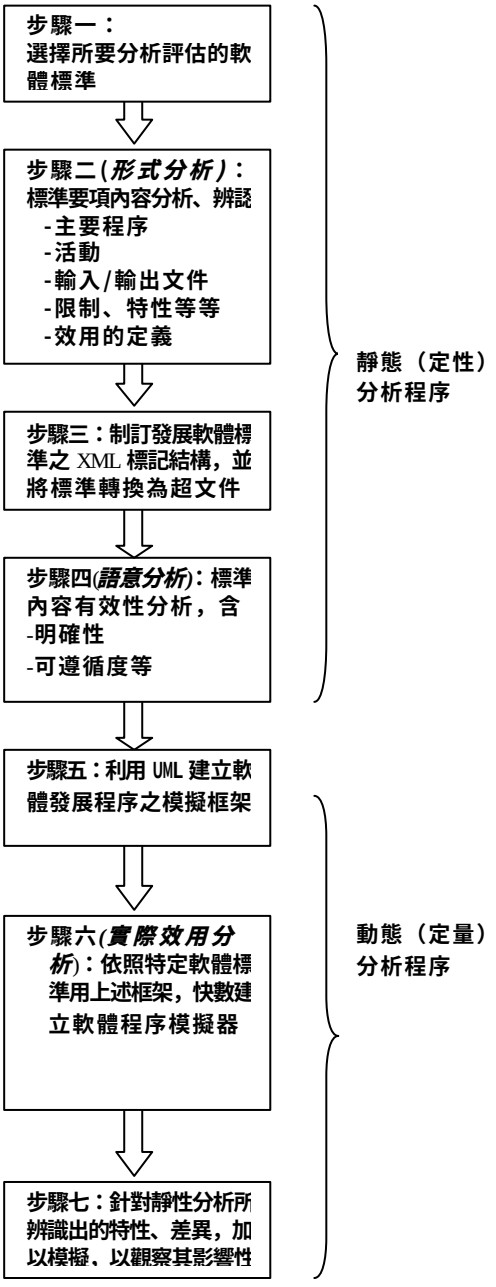
步驟五：

制訂以 UML 為基礎的軟體發展程序框架。

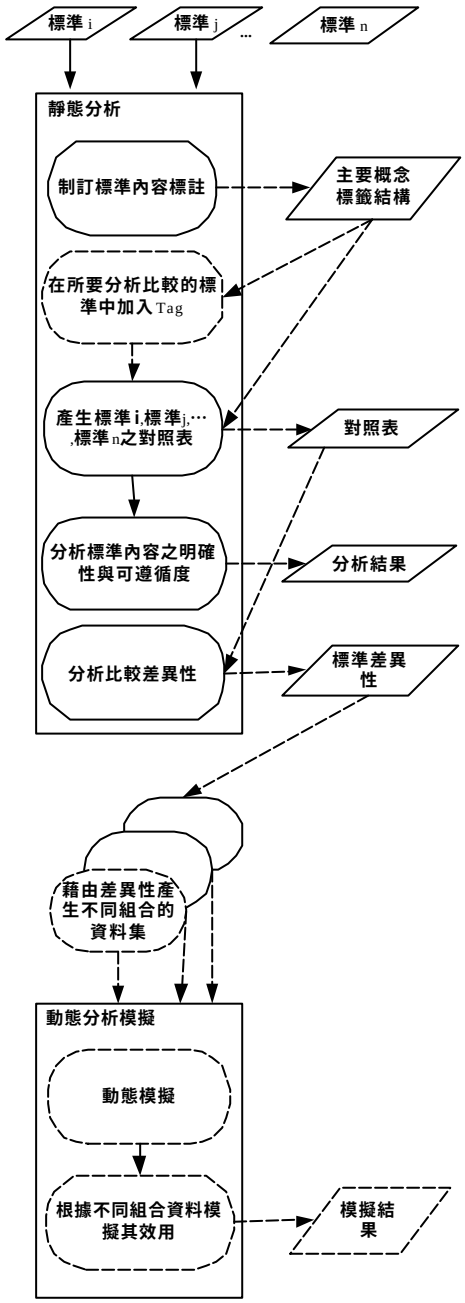
步驟六及七：

利用上述的模擬框架針對特定標準快速建構軟體發展程序的模擬器，以了解應用上不同標準或特定議題的實際效用(pragmatic effects)。

靜態分析與動態模擬分析部分的關聯架構圖如圖二中所示。



圖一　分析方法建立程序



圖二　靜態分析與動態分析之關聯架構圖

四、以 UML 為基礎的模擬框架 (Framework)

本研究希望利用靜態分析的結果來進行動態的模擬，以得到實際效用的數據。然而軟體程序標準眾多，軟體發展模式及技術亦大不相同，沒有一個單一的模擬程式可適用所有不同的需求。然而這些軟體程序模擬器在設計上具有相當大的雷同處。本研究首先發展出可重複使用的軟體程序模擬框架。框架為針對相似的問題所提出通用的設計方法。為我們改良 Young Jong Yang[11]的方法建構以 UML 為基礎的物件導向框架。

因為模擬為內部觸發的事件。我們所提出的框架發展方法如下所述：

(1) 設計使用案例圖(use case)
(2) 藉由每個使用案例觸發的內部事件畫出每個使用案例的活動圖(activity diagram)。
(3) 依照上述活動圖的特性將活動圖分割為數個框架(frameworks)。
(4) 對於每個定義的框架，設計其類別(class)與其類別間的關係。
(5) 如果在框架中具有一個以上的繼承類別，則透過循序圖(sequence diagram)來指定在框架中的類別間的控制流程。
(6) 設計每個框架的介面(interface)。
(7) 在最上層定義所有在框架間的互動。

軟體程序模擬主要動作為內部的活動，如圖三所示，在圖中的實線表示活動的流程，而虛線則表示資料的參考。我們將圖三的三個使用案例(use case)的內部活動分為五個物件導向的框架：(1)程序框架（Process Framework)、(2) 資源框架（Resource Framework)、(3) 產品框架（Product Framework)、(4)風險框架（Risk Framework)、(5)GUI 框架（GUI Framework)。分述於下節。而每一個框架圖中可包含了類別圖、框架與外界間的介面和描繪在此框架中類別間的控制流程的循序圖。

4.1 程序框架

我們根據 IEEE/EIA12207，在程序框架（Process Framework）中，我們定義標準的軟體生命週期中基本的發展與驗證活動，並且在做驗證部分的活動乃是參照 IEEE 1012，如圖四所示。而在本研究中是藉由發展計畫（Development Plan）去選擇與控制所會進行的活動與工作，故可適用於瀑布式(waterfall)或 OO 等不同模式。

在程序框架中，典型的資料輸入可能包括：

(1) 程序/活動的選擇(工作相依性、執行的順序與執行時間)
(2) 專案的屬性(型態、大小與複雜度)

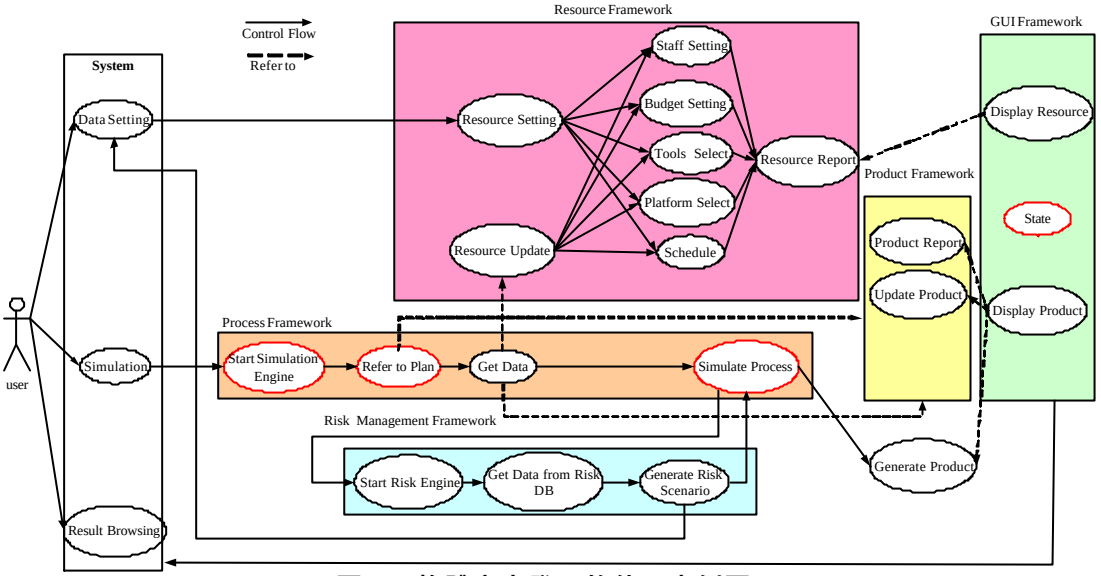
而典型的輸出則包括：

(1) 程序的狀態(階段與完成度)

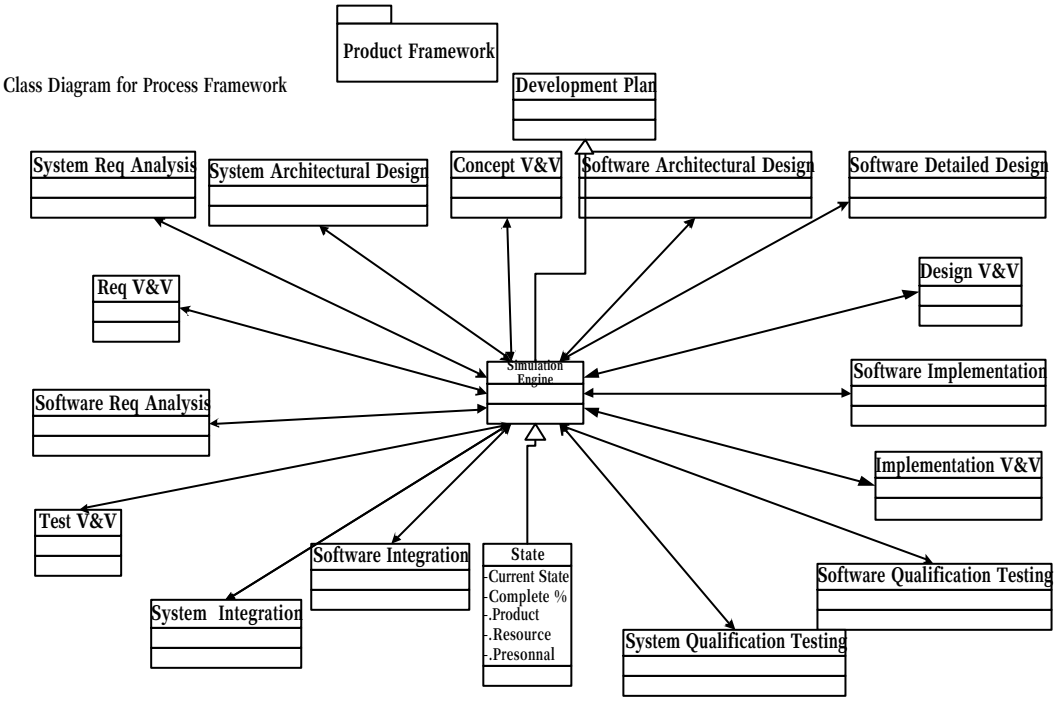
4.2 資源框架

在軟體發展的過程中，包含時間/排程、人力、預算、工具與執行平台，所以在資源框架（Resource Framework）中，則定義了多種不同的資源類別，如圖五所示。

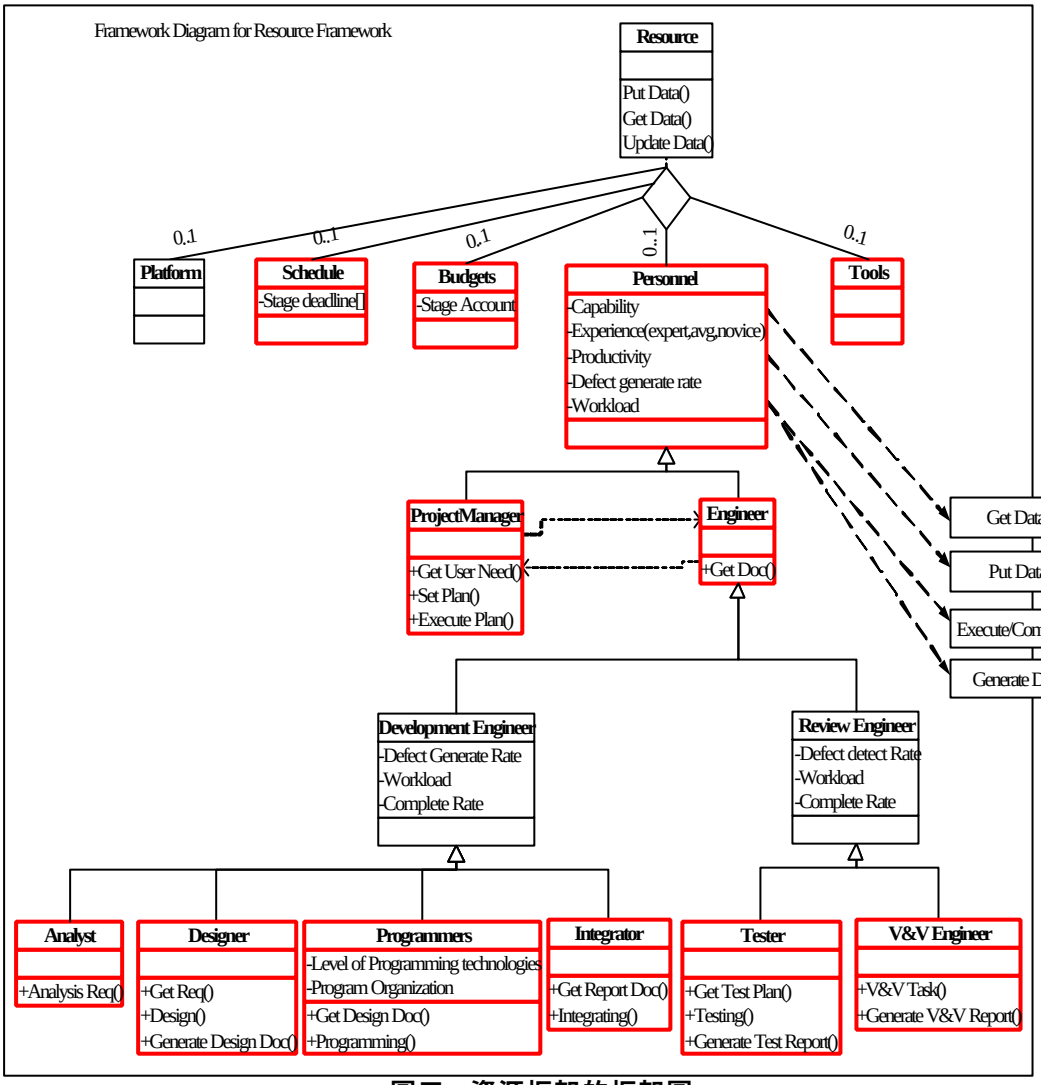
我們將發展者（Developer）分為兩種人員類別，專案管理者（Project Manager）與工程師（Engineer），此兩種人員會繼承發展者的類別。程師又分為分析師、設計者、程式撰寫人員及測試人員等等。



圖三　軟體專案發展的使用案例圖



圖四　程序框架的物件類別圖



圖五　資源框架的框架圖

此框架所需資源相關的輸入資料包含有：

(1)**排程資料**

(2)**人員數目**

(3)**每個人員的能力與經驗值**(**專家、一般人員或新手**)

(4)**不同經驗值人員的生產力**

(5)**不同經驗值人員的錯誤產生率**

(6)**不同經驗值人員的錯誤移除率**

輸出資料則有：

(1)**目前資源的狀態**

4.3 風險框架

由於在軟體的發展過程中可能會有一些突發情況發生，所以程序模擬就必須去考慮到這些事件的發生。風險框架（Risk Framework**）包含使用者的設定（**data Set by the user**）、風險因素資料庫（**risk factor DB**）以及風險產生引擎，如圖六所示。**

潛在的風險事件包含有：

(1)**人員的更換**

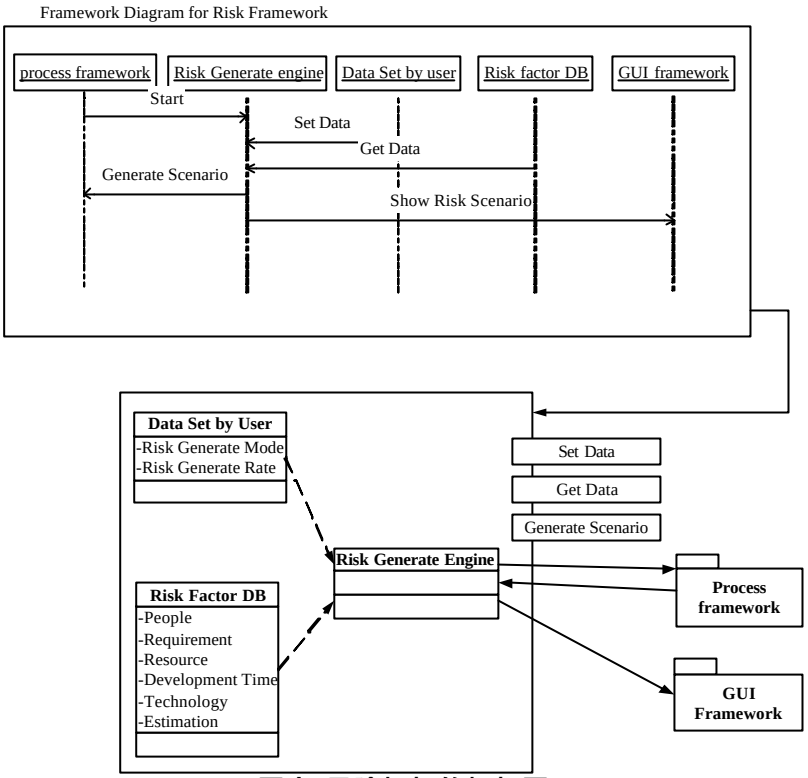
(2)**不適用的硬體**

(3)**需求的改變**

(4)**大小估計不足**

(5)**低效率的** CASE **工具**

風險管理的對策則可預先決定、先存放在資



圖六 風險框架的框架圖

料庫中或是在執行時，使用者由互動模式來介入處理。圖中所示風險框架將可與 GUI **框架及程序框架間有許多的互動。風險框架包含了完整的類別圖與外界的界面及類別間控制的循環圖。**

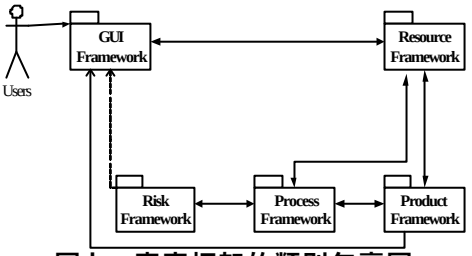
4.4 產品框架及圖形框架

此外，在產品框架（Production Framework**）中，則將在軟體發展過程中所會產生的各項文件定義出來，如需求規格書、設計書、程式碼、測試資料等等。根據** IEEE/EIA 12207.1**，我們將這些文件分為六種不同的型態，並且定義為說明書、計畫、程序、紀錄、報告與規格書。我們亦定義了文件型態相關的屬性及操作方法。**

在圖形介面框架（GUI Framework**）中透過所列圖形介面，來進行專案模擬，給予使用者依不同需求而做出不同的輸入、輸出選擇，如文字（**Text**）、表格（**Table**）與圖形（**Graph**）。而圖形還可依使用者的需求選擇為曲線圖（**Curve**）、圓形圖（**Pie**）、長條圖（**Bar**）等等。**

在專案發展中所包含的每一個框架都定義好並將其整合過後，便可以得到一個完整的專案發展的框架及各框架間之互動圖，如下圖七所示。

我們根據上述框架針對 IEEE/EIA 12207 **及** IEEE 1012 **快速發展一套軟體程序模擬平台，並將其命名為** SimSoft**。**



圖七 專案框架的類別包裹圖

五、案例分析

為驗證本方法的有效性，我們選擇標準比較時，選擇性質相近，但內容仍具有相當差異性的。因為我們的研究計畫所需要(NSC 90-NU-7-155-001**)，所以我們以新舊版** IEEE 1012(1986 **年及** 1996 **年版)為案例。除了上述準則外，亦有下述原因。一般而言在新舊版中對相同的要求項目，新版的標準一定較完善。然而若在要求項目不同時，我們發現下列的問題：**

1. **在新版標準中新增加的議題，對於所要進行的專案發展是否為必須的？**
2. **在新版之前訂的合約或標準大多是引用舊版的標準，在新版標準產生後這些**

專案若仍在進行，是否有必要依新版的標準而改變？

3. 就成本上的考量，遵循新版標準是否合適、有效(cost-effective)？

就 IEEE1012 而言，新增的部份如：軟體安全等級（Software Integrity Level）與 IV＆V 等等的議題，應分析其影響效益。就長期專案(如核四)而言，若訂約時用的是 86 年版本，是否有落伍(out of date)之虞？以下詳述我們以此案例用上述整合靜態、動態分析所做的結果。

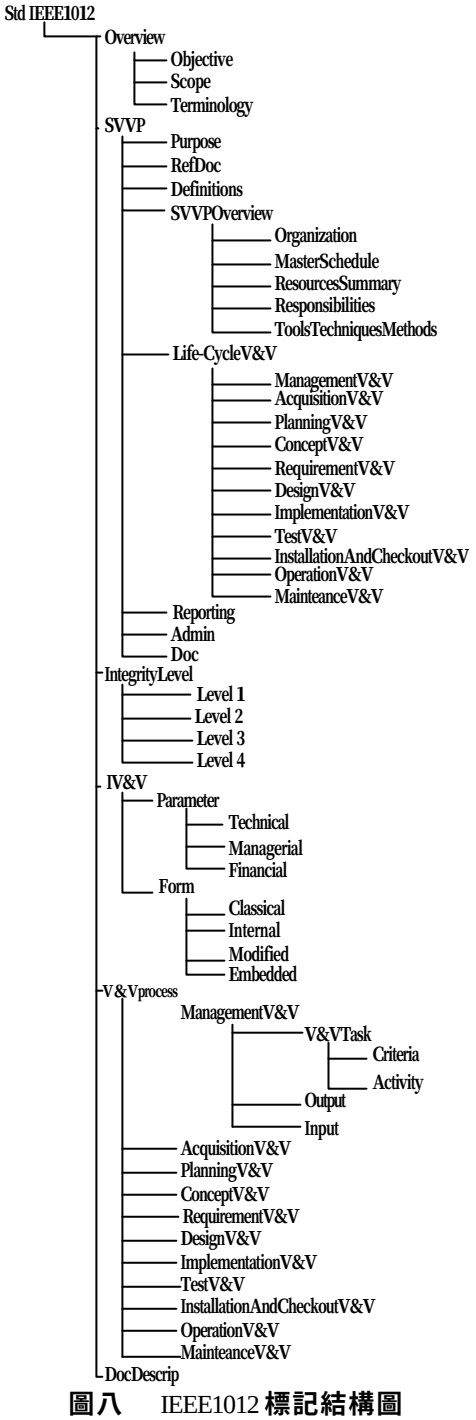
5.1 靜態案例分析

就靜態分析來說，我們首先製定超文件的標記，如圖八所示。新舊版 IEEE 1012 經過此標記結構加註(tag)後，並將內文萃取，便可製成標準差異性與特定要項之對照表。表 1 為 IEEE 1012 -1998 新版中新增的內容。舊版的 IEEE1012-1986 是一份產品的標準，主要在於定義軟體驗證與確認計畫的內容（ SVVP），而新版的 IEEE 1012 則為一份程序標準，清楚的定義在驗證與確認的程序中所規定的活動以及其相關的工作。主要由新舊版目錄中即可得知新版的 IEEE 1012 所新增的部分內容如下：

1. 軟體的安全等級（Software Integrity Level），
2. 獨立驗證與確認（IV＆V），
3. 新版的標準為一程序標準，舊版的標準為一產品標準，
4. V＆V 指標(metrics)的例子。

這些新增與差異部分將放入下階段模擬的考慮中，藉由觀察這些差異對專案發展的影響。

接著我們就明確性及可遵循度進行分析。就標準的可遵循度來說，舊版的可遵循部分就明顯的不足。舊版的標準當中，對許多的 V＆V 的工作項目裡的需求定義部分，並未明確的說明。如表 2 為一例子。我們可以得知新版的標準較舊版容易去遵循與引用。



圖八　IEEE1012 標記結構圖

表 1 IEEE 1012 -1998 新版中新增的內容

Markup	1998 版	1986 版
Software integrity levels	4.1 Software integrity levels	Null
Software integrity levels	Annex B A Software integrity level scheme	Null
IV&V	Annex C Definition of independent verification and validation	Null
Acquisition	5.2 Process: Acquisition	Null
Acquisition Support V&V	5.2.1 Activity: Acquisition Support V&V	Null
Supply	5.3 Process: Supply	Null
Planning V&V	5.3.1 Activity: Planning V&V	Null

表 2 新舊版標準差異明確性及可遵循度比較

Markup	IEEE 1012-1998	IEEE 1012-1986
Implementation V&V	Interface Analysis. ……..The task criteria are as follows: (3.1) **Correctness** a. Validate the external and internal software interface code in the context of system requirements. (3.2) **Consistency** a. Verify that the interface code is consistent between source code components and to external interfaces (i.e.. hardware, user, operator, and other software). (3.3) **Completeness** a. Verify that each interface is described -and includes ……..… (3.4) **Accuracy ……** (3.5) **Testability…….**	Source Code Interface Analysis. Evaluate …..…for **correctness, consistency, completeness,** and **accuracy**. At a minimum, analyze data items at each interface.

5.2 軟體程序模擬

本研究進行模擬的方式就是將前述靜態分析所得的主要差異性(IV&V 與 Integrity Level)，由模擬來進一步探討。首先利用 UML 所建構的框架實際實作出一套模擬程式。模擬主要是透過觀察專案發展的資源調配，如：專案所會花費的人力成本與資源成本，專案大小（功能函式的數目）等等，針對在不同的資源與條件，進行專案發展的模擬。而在每個模擬階段中所執行不同的工作即為一個函式，而函式中即針對專案發展中所會產生的生產力與錯誤數目進行計算，每個階段皆因型態的不同而產生不同的錯誤數目並耗費成本。在互動式模式的模擬中並對風險方面進行動態的模擬，採用隨機的方式產生不同的風險劇情，並利用所舉出的風險對策來進行專案發展風險部分的模擬。

我們藉由前面所述的模擬框架快速發展出我們的模擬程式。例如在圖九為資源框架中人員類別的宣告部分，我們可以發現各種不同型態的人員都繼承上層工程師所應具有的屬性，並且具有自己各自的特性。藉由實作前面所定義的框架並插入合適的模擬函式，即可快速建構所需要的模擬程式。

模擬函式部份，我們所採用參照 Chi Y. Lin[8]所提出的函式並加以修正與調整，而模擬中所採用的數據則是參考[6] 與[7]。在模擬中所採用的相關公式說明如下：

生產力的計算：

$$P(t-dt,t)=[\sum Si*Pi]*C*L*F \quad \cdots\cdots\cdots\cdots\cdots\cdots (1)$$

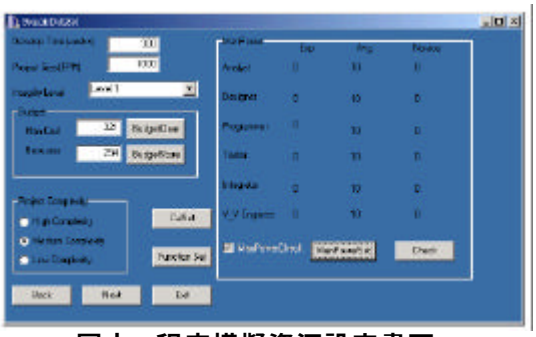
$$C(t)=\{1-t(S)\} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (2)$$

$$t(S)=1-\{1.03\exp(-0.02S)\} \quad \cdots\cdots\cdots\cdots\cdots\cdots (3)$$

在公式（1） 中，可以得到小組單位時間內的生產力為 Si 人員數乘上每種人員的個別生產力 Pi，再乘上在人員內在溝通的因素 C，再乘上學習因素 L 與每單位時間工作完成度 F。其中學習因素與工作完成度在本模擬中我們是藉由人員的經驗與能力來設定。在公式（1）中的溝通因素 C 即為公式（2） 中所計算得到的值。而公式（2） 與（3） 即為人員的內在溝通因素，其中 S 表示人員數目，$t(S)$ 為藉由公式可以得知為指數函式，其圖形為一曲線，到某數量後，人員愈多反而造成生產力的下降，溝通因素 C 即為影響團隊生產力的比率，為內在溝通花費的因素。在公式中藉由考量人員內在的溝通所會造成相關的生產力損失。

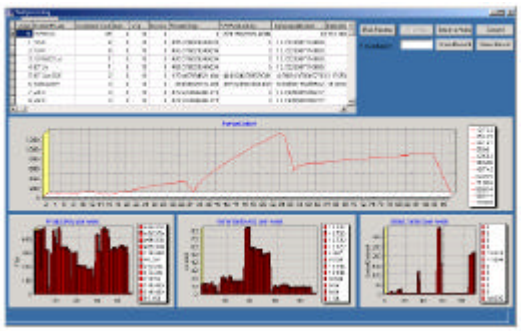
```
//----------manclass----------
class Engineer {
public:
int capability;
int level;              //the Experience of
Engineer
float productivity;
float defectrate;   //the defect generate rate in
the project
float workload;         //the engineer's workload
in the project
int staffnumber;
float removedefectrate;
float completerate; //the complete rate in the task
of project
void getdoc();
private:
};

//analyst
class Analyst : public Engineer{
public:
analysisreq();
private:
};
//designner
class Designner : public Engineer{
public:
getreq();
design();
generatedesigndoc();
private:
};
//programmer
class Programmer : public Engineer{
public:
int technologylevel;
int organization;
void getdesigndoc();
void programming();
```

圖九　人員類別相關實作部分程式碼



圖十　程序模擬資源設定畫面

圖十為部份的輸入畫面，圖十一為典型的輸出畫面。



圖十一　軟體程序模擬過程畫面

錯誤產生數目的計算：

$$E = [\sum Si * Ea * N * Y * M] + [P * U * Z] \cdots\cdots\cdots (4)$$

$$M_{(Mix)} = 2 - Staff(\exp erienced) / Staff(total) \cdots\cdots (5)$$

Y

$$(Schedule Pressure) = 1/4 remainworkate^2 + 1/5 remainworkate + 0.95$$

$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (6)$$

而公式（4） 為計算單位時間內的缺失產生率，其值等於小組單位時間內的生產力為 Si 人員數乘上每種人員的個別缺失產生率 Ea 再乘上其工作缺失量 N 與時程壓力因素 Y和人力混合因素 M 的乘積再加上小組生產力乘上不可偵測的缺失密度 U（t）與因階段改變所產生的異變數 Z。本研究中將不針對不可偵測的缺失密度部分去做考量。

錯誤移除數目的計算：

$$D = \sum Si * Di * M * (1/Y) \cdots\cdots\cdots\cdots\cdots\cdots (7)$$

在公式（7）中，缺失偵測率 D 為我們將 Chi Y. Lin[8]所提出的函式修改，利用不同經驗人員的檢查力乘以人員的數目，再乘上每種人員的缺失檢查量，而在乘上人員混合因素 M 與時程壓力 Y 的倒數。而在本模擬中主要是以人員的工作能力與經驗值來做考慮。

我們在實際模擬時，即利用到上述的功能函式進行模擬計算，而其中所需要得到的主要輸入值為：

-專案的大小（Project Size）
-專案的型態（Project Complexity）
-軟體安全等級（Software Integrity Level）
-人員的生產力（Productivity）
-專案發展人員個數
-各階段的排程（Schedule）

經過模擬之後所要得到的主要輸出值為：

-錯誤數目
-完成時間
-錯誤變化量
-人員的變化量

我們強調的不在於模擬結果的數值，而是在於相同情況下，採用不同議題或不同依據下模擬出來的趨勢比較。而我們設計以功能點（Function Point）為主，參考 Capers Jones[6]所提供的數值，在模擬過後我們藉由觀察模擬的錯誤數目與變化量，資源的消耗與預算的比較及完成工作的總天數，以分析所得到模擬的數據。

5.3　動態模擬案例結果與分析

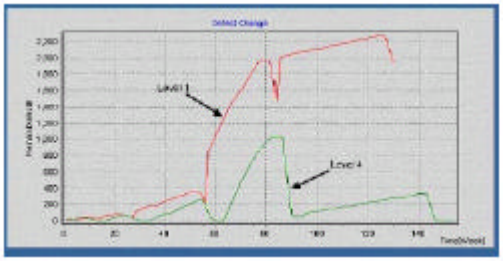
前述靜態分析結果，IEEE1012 新舊版主要差異為 IV&V 及 Integrity Level 就此我們以模擬觀察其效用。以下為我們將對案例進行描述。

案例一：軟體安全等級（Software Integrity Level）

案例說明：

透過相同的發展條件，進行專案的模擬，由於在舊版中並沒有軟體安全等級（Software Integrity Level）的觀念，所以我們以軟體安全等級為一的代表舊版的 IEEE 1012 的驗證與確認的工作標準，與新版 IEEE 1012 程度四相比。

而我們由圖十二中可以得知，程度為一的較程度為四的完成時間早，但是所留有的錯誤數較多。

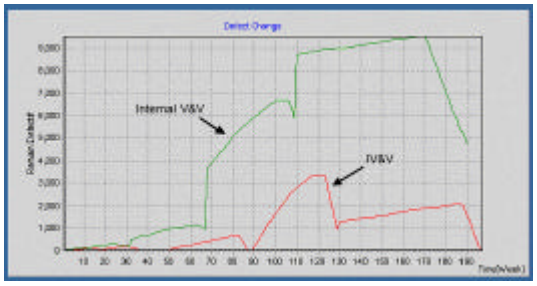


圖十二 Integrity Level 1 vs. Level 4

案例二：內部驗證與確認（Inter V&V）與 獨立驗證與確認（IV＆V）

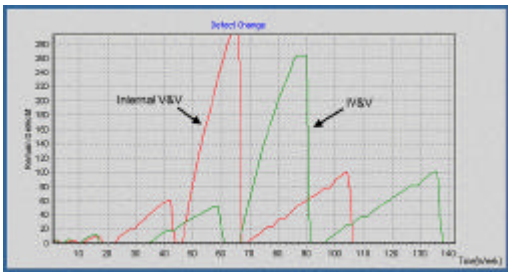
案例說明：

藉由 Internal V& V **與** IV& V **的案例來看，在下列的情況下我們建議採用** IV& V**：發展新型專案，發展者為新手或一般人員，由於原來發展時所產生的缺失數目很多，即原始軟體品質較差時，可用有經驗的** IV& V **人員進行，將會得到較佳的效果，如圖十三所示。**



圖十三 V& V **一般人員** vs. IV&V **專家**

在下面的情況下則不建議採用 IV& V**：當原始軟體品質較佳，即原來發展時所會產生的缺失數目不多，而內部** V&V**人員為專家或一般人員時，並不建議採用** IV& V**，如圖十四。**



圖十四 V& V **專家** vs. IV&V **一般人員**

由上實驗可知模擬將眾多因素及實際情境(scenarios)**組合下，不同狀況同議題可有不同的結果，故標準新加的要求是否符合成本效益**(cost effectiveness)**，例如** IV&V **議題，結論依情況而定。如此整合靜態、動態的準分析方法，將形式、內容及實際效用層的因素都完整地考慮進去。**

六、 結論

本研究主要目的是對軟體工程技術標準實際應用有效性提供一套評估方式，而這樣的方式包含了靜態與動態的分析方式，而所提供以下幾項技術：

1. **本研究中靜態部分採取** XML**超文件的方式並將標準的結構重建，並透過萃取標準的內容與資料庫檢索的方式，很快的找到所要進行分析與比較的標準內容，並產生比較的對照表，可增加發展者與分析者在執行上的效率。**

2. **發展一以** UML **為發展基礎具通用性並可重複使用性的軟體程序模擬框架。可容易被重複利用於實作的軟體程序模擬。**

3. **發展出一軟體程序模擬程式，可藉由調整軟體發展中的各項因素，觀察影響軟體專案品質的績效與其相互關係。**

此套方法中，靜態分析可顯示標準形式(syntactic)**上的特質或差異，亦有助於了解內容語意上**(semantic)**可能之效用。動態模擬則提供實際效益**(pragmatic)**上的分析，此為一套有系統的分法。不僅僅是適用於本研究中的案例** IEEE 1012**，亦可以在評量其他不同標準的效用。**

誌謝

本研究由國科會及原能會贊助計劃編號 NSC 90-NU-7-155-001.

七、 參考文獻

[1] James D. Arthur, "Evaluating the Effectiveness of Independent Verification and Validation," IEEE Computer, 1999

[2] Norman E. Fenton and Martin Neil, "A Strategy for Improving Safety Related Software Engineering Standards," IEEE transactions on software engineering, vol. 24, no. 11,November 1998

[3] Debra Herrmann, "A Methodology for Evaluating, Comparing and Selecting Software Safety and Reliability Standards," IEEE COMPASS, pp 223-231, 1995

[4] Dirk Jager, Ansgar Schleicher, Bernhard Westfechtel, "Using UML for Software Process Modeling," 7th European Software Engineering Conference, Toulouse, France, Sep, 1999.

[5] R. E. Johnson, and B. Foote, "Designing Reusable Classes," Journal of Object-oriented Programming, 1(2), June 1988.

[6] Capers Jones, "Software benchmarking," Software Productivity Research, Oct, 1995.

[7] Shinji Kusumoto , "A New Software Project Simulator Based on Generalized Stochastic Petri-net," Proceeding of 19th International Conference on Software Engineering, pp 293-302, 1997.

[8] Chi Y. Lin, "Software -Engineering Process Simulation Model (SEPS)", *J. SYSTEM SOFTWARE* , 38,pp.263-277, 1997.

[9] S. Pfleeger, N. Fenton, and S. page, "Evaluating Software Engineering Standards," IEEE Computer, Sept. 1994, pp 71-79.

[10] Ian Somerville and Stephen Viller, "Safety-critical System Programme Process Comparison", http://www.aber.ac.uk/~dcswww/SCSP/reports/lancs.htm.

[11] Young Jong Yang, Soon Yong kim and Gui Ja Choi,, "A UML-based Object-Oriented Framework Development methodology," 1998 Proceedings of Asian Pacific Software Engineering Conference, 1998, pp 211-218.