軟體可靠度模型及失效分析

楊尚青[1]　郭壽齡[2*]

[1]國防大學管理學院資源管理及決策所
[2]國防部聯合後勤司令部兵工整備發展中心

摘　要

軟體可靠度成長模型(Software Reliability Growth Models, SRGMs)之理論發展自 1970 年代開始，相關研究對軟體可靠度模型及預測效能已有相當之探討，並提出一些新的模型；雖然目前相關數學模型及分析步驟已經具備，唯其中假設事項及參數頗多，且各學者專家所提出之模型類型太多，在應用時容易產生混淆，仍有待整合出軟體可靠度的通用模型，以利實際應用及計算。韋伯分配是以指數分配為基礎推導而得到比較符合軟體實際失效狀況的機率分配模型，本研究以非齊性的類似更新韋伯過程(Nonhomogeneous Quasi-Renewal Weibull Process, NQRWP)建構軟體可靠度通用模型，藉由韋伯分配的形狀參數及軟體可靠度成長參數之改變，得到軟體失效模式之 9 種狀況，可以模擬及預測軟體失效行為模式。

關鍵詞：類似更新，軟體可靠度，韋伯分配

A Study of Software Reliability Model and Failure Analysis

Sang-Chin Yang[1], Shou-Ling Kuo[2*]

[1]*Resources Management and Decision Institute, Management College, National Defense University, R.O.C.*
[2]*Ordnance Readiness Development Center, Combined Logistics Command, Ministry of National Defense, R.O.C.*

ABSTRACT

The theorem of software reliability models has been developed since 1970s. Many researchers have discussed about software reliability models, and have advocated some new models. Although the existing models have been possessed of mathematical methods and analytical processes, their assumptions and associated parameters are not practical and difficult to implement. Under the assumption of Nonhomogeneous Quasi-Renewal Weibull Process (NQRWP), this paper has presented a general software reliability model for practical applications. Changing the shape parameter of Weibull distribution and the software reliability growth parameter, this paper has categorized nine conditions of software failure behavior. The results can be applied to software failure simulation and prediction.

Keywords: quasi-renewal, software reliability, weibull distribution

文稿收件日期 98.5.26;文稿修正後接受日期 99.11.30;*通訊作者
Manuscript received May 26, 2009; revised November 30, 2010; * Corresponding author

一、前　言

科學技術的進步使系統設備產生了重大的變革，重要設施（如核電廠）、交通運輸系統（如高鐵、捷運、飛機）及大型武器系統（如C4ISR）等新型系統的使用大幅提高了便利性及使用效能，自動化、智慧化、高速度是新時代系統的基本特徵。由於系統規模愈來愈複雜，純粹由人力操控已無法滿足系統的反應及要求，必須藉由軟體的運作執行才能夠發揮系統優異的效能，因此軟體運用於系統之操作執行已為必然，系統對於軟體可靠度的要求也愈來愈嚴格[1,2]。軟體可靠度行為模式的充分瞭解可以提昇軟體的執行效能及安全，並可建立軟體與硬體間之關係或影響，以減低重大災難發生的機率[3]。尤其是 21 世紀資訊工業的迅速發展，軟體發展時程也愈來愈短，因此在軟體系統生命週期各階段中，我們應善用適當軟體可靠度工程的技術與工具，以保證系統運作正常，安全可靠。

軟體可靠度模型之理論自 1970 年代開始發展[1]，相關之研究如 Choi and Seong[4]、Levitin[5]、Gokhale[6] 、Teng and Pham[7]及Huang and Lyu[8]等，對軟體可靠度模型及預測效能均有相當之探討，並提出一些新的模型；雖然目前相關數學模型及分析步驟已經具備，唯其中假設事項及參數頗多，且各學者專家所提出之模型類型太多，在應用時容易產生混淆，仍有待整合出軟體可靠度的通用模型，以利實際應用及計算。

軟體不可能單獨存在，任何軟體都必須藉由硬體才可以執行及顯示結果，故本論文只考慮軟體本身程式錯誤或缺點及軟體與硬體不相容或衝突所造成之失效，對於硬體失效而造成軟體無法執行的狀況，則視為軟體正常。雖然 Vonta et al.[9]認為軟體可靠度與硬體可靠度基本上是不同的，軟體失效是由於不完美(imperfect)的程式碼所造成，而物質惡化則為硬體失效的主要的原因；因此技術的發展能改進硬體的可靠度，但軟體則不行。而且，硬體可以重複設置造成冗餘性(redundancy)以提高可靠度，但是軟體程式複製後具有完全相同的缺陷，無法提高軟體可靠度。然而 Lyu[10]指出軟體可靠度與硬體可靠度相類似，兩者都屬於隨機過程(stochastic process)，而且都可以機率分配來描述。軟體雖然不會隨著時間產生損耗或惡化，但是因為軟體的失效發生後其錯誤或缺點能被移除，因此軟體的可靠度會有所成長（逐漸變好）；另一方面，軟體使用方式或操作習慣的改變，以及不當的修改，亦有可能使軟體可靠度變差。

Blischke and Murthy[11]則主張雖然基本上軟體可靠度與硬體可靠度不同，通常硬體會隨著時間產生損耗，而軟體可經由除錯隨著時間提高可靠度； 但是也有例外，例如硬體可經由改良而提高可靠度，而軟體可能會因為發現新的程式錯誤(bugs)使可靠度減低。此外，Pham[12]也認為軟體在生命週期各階段可能因不斷的修改而具有不同的可靠度模型，與其他軟體或硬體共同使用時亦可能互為影響，造成適用模型變更。

Musa[13]則將軟體可靠度模型區分為兩種，其一為「錯誤未移除(no fault removal)」，通常在確認測試(certification test)或實地操作測試(operation in the field)時使用；另一種為「錯誤移除(with fault removal)」，通常在可靠度成長測試(reliability growth test)時使用。而所謂的「錯誤未移除(no fault removal)」實際上是指「錯誤延期移除(deferred fault removal)」，測試者確認錯誤後並沒有立即移除錯誤，而是在下一次發行(release)時才移除，這與目前一般軟體以不定期線上更新之模式相符，也是本論文所要探討的軟體可靠度模型。

由於各學者專家提出的軟體可靠度模型很多，不同的觀點在分類上會有所不同，例如時間模型或狀態模型、連續模型或不連續模型、缺點數有限模型或缺點數無限模型，因此在模型的分類上也見仁見智。Musa and Okumoto[14]以五個屬性分類，分別為(a)Time domain，日曆時間或執行時間 (b)Category，失效次數有限或失效次數無限 (c)Type，失效次數經驗值於時間上之分佈 (d)Class，以時間為變數之失效密度函數 (e)Family，以失效次數期望值為變數之失效密度函數。Xie[15]將軟體可靠度模型區分為馬可夫模型(Markov models)、非齊性卜瓦松過程模型(NHPP models)、貝氏模型(Bayesian models)、統計資料分析模型(Statistical data analysis models)、操作輸入模型(Input-domain-based models)、注入

標籤模型(Seeding and tagging models)、軟體結構模型(Software metrics models)等七類。Blischke and Murthy[16]及 Ledoux[17]則認為軟體可靠度模型可以白箱(white-box)及黑箱(black-box)分類，白箱模型對於程式碼的複雜程序及結構進行分析，以評估軟體的可靠度；黑箱模型則以執行的頻率或失效時間間隔做外部的評估，與機率分配函數有關。本研究以建模方式為區分，將軟體可靠度模型概分為三類，分別為：

(1) 時間機率模型：非齊性卜瓦松過程(Nonhomogeneous Poisson Process, NHPP)[4,6-8,27,28,30-36,39,40]；
(2) 狀態機率模型：馬可夫過程(Markov Process, MP)[3,18-20,41]；以及
(3) 實際資料統計分析模型：經由重複實驗或故障樹分析(Fault Tree Analysis, FTA)而得到之實際值[21-25,42]。

Ledoux[43]認為雖然軟體的失效是由於程式設計錯誤所引起的，因此不是一個隨機現象；但是因為必須經由輸入的動作才會發現軟體失效，而輸入的動作是一個隨機過程，故可以隨機過程來描述軟體的失效行為。雖然 Washburn[26]認為應該先觀察軟體失效及修復行為是否符合非齊性卜瓦松過程，不宜一開始就假設為非齊性卜瓦松過程；然而因假設非齊性卜瓦松過程可以簡化計算達到模擬及預測軟體失效及除錯行為之目的，故已廣為相關研究者所接受[30-33,44]。Gokhale[6]、Okamura et al.[27]、Kuo et al.[28]、Malaya et al.[29]及 Zhang and Pham[45]等人雖已將幾種不同的機率分配模型進行比較分析；但是在各種不同的機率分配中，如何選擇適當的機率分配還是讓人困擾的問題。許多學者如 Choi and Seong[4]、Levitin[5,36,37]、Teng and Pham[7]、Lawson et al.[35]等亦提出軟體不同版本(version)的可靠度成長模型，唯各模型差異頗大，使人有無所適從之感。

軟體失效模型與硬體失效模型相類似，但是軟體修復與硬體修復狀況則有所不同。硬體修復後之可靠度模型有通常有兩種狀況，也就是和新的一樣(完全更新)或比原來的差(最小修理與完全更新之間)；而軟體除錯後之可靠度模型有三種狀況，分別為比原來好（錯誤完全或部分移除，且沒有因修改程式而發生新的錯誤）、和原來一樣（錯誤沒有移除，或錯誤部分移除但是因修改程式而發生新的錯誤）及比原來更差（錯誤沒有移除或部分移除，但是因修改程式而發生新的錯誤比原來更嚴重）。依據我們計算硬體可靠度的經驗[46]及 Kuo et al.[28]、Liu et al.[34]、Ravishanker et al.[47]所建議，韋伯分配以指數分配作為其特殊型式，可推導而得到比較符合軟體實際失效狀況的機率分配模型。當其形狀參數 β (shape parameter)小於 1 時，失效率隨著使用時間而降低，為失效率遞減型(Decreasing Failure Rate, DFR)；當 β 等於 1 時，失效率保持一常數，不隨著使用時間而改變降低，為常數失效率型（Constant Failure Rate, CFR），也就是指數分配模型；當 β 大於 1 時，失效率隨著使用時間而升高，為失效率遞增型(Increasing Failure Rate, IFR)。

Tkachtenberg[38]於 1990 年提出軟體可靠度通用模型，指出軟體的失效率(failure rate)決定於軟體的「平均錯誤程度(average error size)」、「發生錯誤密度(apparent error density)」和「工作負荷(workload)」，並探討瞬時可靠度各係數與初始可靠度係數間之關係。雖然 Tkachtenberg 之可靠度通用模型已具有模型單一化的特性，但在軟體規模愈來愈大，結構愈趨複雜及資訊技術快速發展的狀況下，有必要依目前軟體技術之使用、失效及除錯行為建構新的模型。因此，本研究提出以非齊性的類似更新韋伯過程 (Nonhomogeneous Quasi-Renewal Weibull Process, NQRWP)建構軟體可靠度之通用模型，藉由形狀參數 β 及成長參數 u 之改變，得到軟體失效模式之 9 種常見狀況，可以模擬及預測軟體失效行為模式。因此，我們只要改變軟體可靠度模型中的相關參數，就可以一個數學式來表示所有軟體失效的狀況，可避免使用不適當的數學模型來計算及評估軟體的可靠度，達到模型單一化之目的。

二、類似更新理論

符號說明

$f_{X_n}(t)$：第 n 次失效的機率密度函數
$F_{X_n}(t)$：第 n 次失效的累積機率函數
$\overline{F}_{X_n}(t)$：第 n 次失效的可靠度函數
$L\{f(t)\}$ 或 $f^*(s)$：$f(t)$ 的 Laplace 變換式
$L^{-1}\{f(t)\}$：$f(t)$ 的 Laplace 逆變換
$Q(t)$：類似更新函數

$Q^*(s)$：類似更新函數的 Laplace 逆變換
$U_{n-1}(n)$：第 n 次失效的可靠度成長函數
u：可靠度成長參數
$(U_1(2)=U_2(3)=\ldots=U_{n-1}(n)=u)$
Ξ_n：前 n 次失效操作時間長度之和

英文縮寫

i.i.d.：相同且獨立的機率分配(Independently and identically distributed)
r.v.('s)：隨機變數(Random variable(s))
Cdf：累積機率函數(Cumulative distribution function)
pdf:機率密度函數(Probability density function)

2.1 類似更新過程(Quasi-renewal process)

Wang and Pham[48, 49]於 1996 年定義類似更新過程為具有可靠度成長參數 u 及第 1 次失效發生的時間長度 X_1 的機率函數程。本研究依據 Wang and Pham 的定義將參數 u 以函數 $U_{n-1}(n)$ 表示，則每次軟體更新後之可靠度成長參數不一定會相同，如此較符合實際狀況：

定義 1：若失效發生為一隨機可數的連續過程 $\{N(t),t\geq 0\}$，在第 $n-1$ 次與第 n 次失效發生的時間長度為 X_n，$n\geq 1$。觀察此連續非負的隨機變數 $\{X_1,X_2,X_3,...\}$，若有一函數 $U_{n-1}(n)$ 使得 $X_1=U_0(1)\cdot Z_1$，$X_2=U_0(1)\cdot U_1(2)\cdot Z_2$，$X_3=U_0(1)\cdot U_1(2)\cdot U_2(3)\cdot Z_3$,..., $X_i=\prod_{n=1}^{i}U_{n-1}(n)Z_i$，其中 Z_i 為相同獨立分配(independent and identically distribution, iid)的隨機變數，而且 $U_0(1)=1$。則稱此連續過程 $\{N(t),t\geq 0\}$ 為具有可靠度成長函數 $U_{n-1}(n)$ 及第 1 次失效發生的時間長度 X_1 的類似更新過程。

根據定義 1，當 $U_1(2)=U_2(3)=...=U_{n-1}(n)=u$ 時，表示每次軟體更新後，其可靠度成長參數均不變，是一個較特殊的情況，但有利於類似更新函數的推導。我們可以第 1 次失效發生時間的 *Cdf*、*pdf* 及失效率(failure rate)來表示軟體失效行為模型之 Cdf、pdf 及失效率如下：

$$F_{X_n}(t)=F_{X_1}\left(u^{-(n-1)}t\right) \tag{1}$$

$$f_{X_n}(t)=\frac{f_{X_1}\left(u^{-(n-1)}t\right)}{u^{n-1}} \tag{2}$$

$$z_{X_n}(t)=\frac{z_{X_1}\left(u^{-(n-1)}t\right)}{u^{n-1}} \tag{3}$$

2.2 類似更新函數

令 Ξ_n 為前 n 次失效操作時間長度之和，$\Xi_n=\sum_{i=1}^{n}X_i$，則類似更新函數 $Q(t)$，或類似更新次數的期望值，可以 $F_{\Xi}(t)$ 的 n 次摺積(n-fold convolution)表示如下：

$$\begin{aligned}Q(t)&=E[N(t)]=\sum_{n=0}^{\infty}n\Pr\{N(t)=n\}\\&=\sum_{n=1}^{\infty}n\left(\Pr\{N(t)\geq n\}-\Pr\{N(t)\geq n+1\}\right)\\&=\sum_{n=1}^{\infty}n\left(F_{\Xi}^{(n)}(t)-F_{\Xi}^{(n+1)}(t)\right)=\sum_{n=1}^{\infty}F_{\Xi}^{(n)}(t)\end{aligned} \tag{4}$$

其中 $F_{\Xi}^{(n)}(t)$ 為 $F_{\Xi}(t)$ 的 n 次摺積。為使類似更新函數 $Q(t)$ 的數學運算易於處理，先求得 $F_{\Xi}^{(n)}(t)$ 的 Laplace 變換式是有幫助的，推導如下：

$$\begin{aligned}F_{\Xi}^{(n)*}(s)&=L\left\{F_{\Xi}^{(n)}(t)\right\}=L\left\{\Pr\left\{\sum_{i=1}^{n}X_i\leq t\right\}\right\}\\&=L\{\Pr\{X_1+X_2+...+X_n\leq t\}\}\\&=\frac{1}{s}f_{\Xi}^{(n)*}(s)\\&=\frac{1}{s}f_{X_1}^{*}(s)f_{X_2}^{*}(s)...f_{X_n}^{*}(s)\\&=\frac{1}{s}f_{X_1}^{*}(s)f_{X_1}^{*}(us)...f_{X_1}^{*}\left(u^{n-1}s\right)\end{aligned} \tag{5}$$

對(4)式等號兩邊取 Laplace 變換式，並將(5)式代入(4)式，則可得惡化(deteriorating, $0<u<1$)或改善(improving, $u>1$)之類似更新函數

$$Q^*(s)=L\left\{\sum_{n=1}^{\infty}F_{\Xi}^{(n)}(t)\right\}=\sum_{n=1}^{\infty}\frac{1}{s}f_{X_1}^{*}(s)f_{X_2}^{*}(s)...f_{X_n}^{*}(s)=\frac{1}{s}\left[f_{X_1}^{*}(s)+f_{X_1}^{*}(s)\sum_{n-1=1}^{\infty}f_{X_2}^{*}(s)f_{X_3}^{*}(s)...f_{X_n}^{*}(s)\right] \tag{6}$$

對(2)式取 Laplace 變換，可得

$$f_{X_n}^*(s)=f_{X_1}^*\left(u^{n-1}s\right)=f_{X_1}^*\left(u^{n-2}us\right)=f_{X_2}^*\left(u^{n-3}us\right)=\cdots=f_{X_{i-1}}^*\left(u^{n-i}us\right) \tag{7}$$

將(7)式代入(6)式可得

$$\begin{aligned}Q^*(s)&=\frac{1}{s}\left[f_{X_1}^*(s)+f_{X_1}^*(s)\sum_{n-1=1}^{\infty}f_{X_2}^*(s)f_{X_3}^*(s)..f_{X_n}^*(s)\right]\\&=\frac{1}{s}\left[f_{X_1}^*(s)+f_{X_1}^*(s)\sum_{n-1=1}^{\infty}f_{X_1}^*(us)f_{X_2}^*(us)..f_{X_{n-1}}^*(us)\right]\\&=\frac{1}{s}\left[f_{X_1}^*(s)+f_{X_1}^*(s)\sum_{n-1=1}^{\infty}f_{X_1}^*(us)f_{X_1}^*\left(u^2s\right)..f_{X_1}^*\left(u^{n-1}s\right)\right]\end{aligned} \tag{8}$$

對(8)式取 Laplace 逆變換，可得以第 1 次失效時間長度 X_1 的 *pdf* 的 Laplace 變換式來表示類似更新函數

$$Q(t)=L^{-1}\left\{Q^*(s)\right\}=L^{-1}\left\{\frac{1}{s}\left[f_{X_1}^*(s)+f_{X_1}^*(s)\sum_{n-1=1}^{\infty}f_{X_1}^*(us)f_{X_1}^*\left(u^2s\right)\cdots f_{X_1}^*\left(u^{n-1}s\right)\right]\right\} \tag{9}$$

有關類似更新函數的數值計算可參考 Rehmert[50]的論文。

三、軟體可靠度模型

考慮某軟體發生第 $n-1$ 次與第 n 次失效的時間長度為連續非負隨機變數 $X_n\in\{X_1,X_2,X_3,...\}$，$n\geq 1$，並服從某機率分配 $F_{X_n}(t)$。假設軟體操作時間符合類似更新過程，第 1 次失效發生的時間長度為 X_1，可靠度成長參數為 u，則我們可以依據軟體失效的狀況，應用非齊性的類似更新韋伯過程來建構軟體可靠度之通用模型。

根據定義 1，當 $U_1(2)=U_2(3)=U_{n-1}(n)=u$ 時，以非齊性類似更新韋伯過程所建構之軟體失效行為模式以第 1 次失效時間長度 X_1 的 *Cdf*、*pdf* 及失效率表示如下：

$$F_{X_n}(t)=F_{X_1}\left(u^{-(n-1)}t\right)=1-e^{-\left(\frac{u^{-(n-1)}t}{\eta}\right)^{\beta}}\quad;\ t\geq 0 \tag{10}$$

$$f_{X_n}(t)=\frac{f_{X_1}\left(u^{-(n-1)}t\right)}{u^{n-1}}=u^{-(n-1)}\frac{\beta}{\eta}\left(\frac{u^{-(n-1)}t}{\eta}\right)^{\beta-1}e^{-\left(\frac{u^{-(n-1)}t}{\eta}\right)^{\beta}}\quad;\ t\geq 0 \tag{11}$$

$$z_{X_n}(t)=\frac{z_{X_1}\left(u^{-(n-1)}t\right)}{u^{n-1}}=u^{-(n-1)}\frac{\beta}{\eta}\left(\frac{u^{-(n-1)}t}{\eta}\right)^{\beta-1}\quad;\ \eta>0,\ \beta>0,\ t\geq 0. \tag{12}$$

軟體失效率函數與時間之關係如圖 1 至圖 9，其中黑色實線為軟體原始失效率函數(n=1)，藍色虛線為第 1 次更新後之失效率函數(n=2)，紅色實線為第 2 次更新後之失效率函數(n=3)，綠色虛線為第 3 次更新後之失效率函數(n=4)。由圖 1 至圖 3 可知當 β<1 時，軟體失效率隨著使用時間而降低，為失效率遞減型；圖 4 至圖 6 顯示當 β=1 時，不論軟體離上次失效時間長短，其失效率在任何時間均相同，屬機遇失效(chance failures)，也就是指數分配模型；而圖 7 至圖 9 表示當 β>1 時，失效率隨著使用時間而升高，為失效率遞增型。不論軟體失效率隨著使用時間而降低、相同或升高，軟體除錯後會有三種狀況，分別為比原來好（錯誤完全或部分移除，且沒有因修改程式而發生新的錯誤，如圖 3、圖 6 及圖 9）、和原來一樣（錯誤沒有移除，或錯誤部分移除但是因修改程式而發生新的錯誤，如圖 2、圖 5 及圖 8）及比原來更差（錯誤沒有移除或部分移除，但是因修改程式而發生新的錯誤比原來更嚴重，如圖 1、圖 4 及圖 7）。

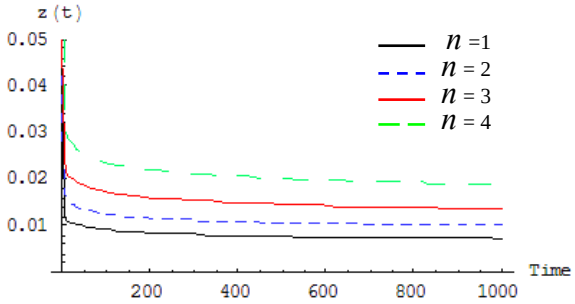

圖 1. 失效率遞減型，但逐漸惡化。

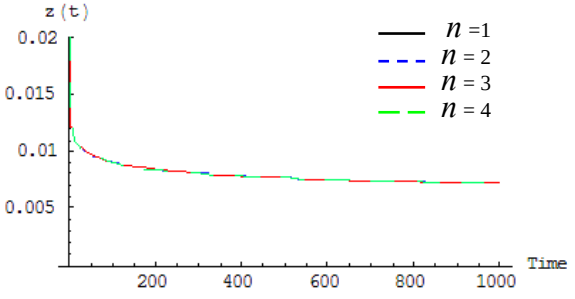

圖 2. 失效率遞減（renewal）。

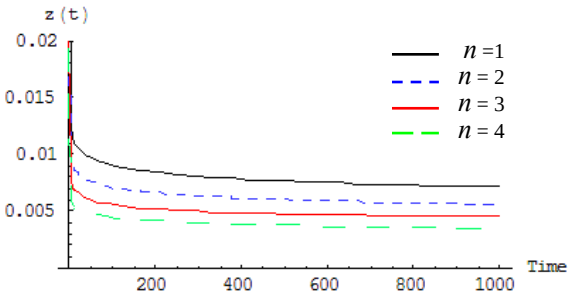

圖 3. 失效率遞減型，且愈來愈好。

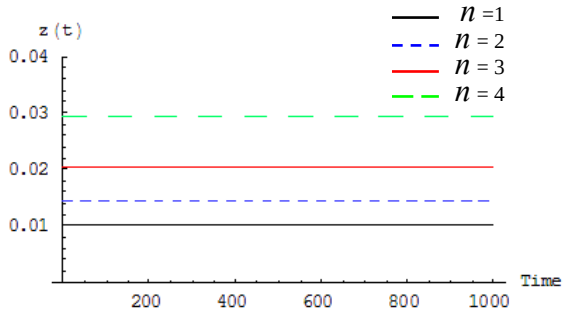

圖 4. 常數失效率型，但逐漸惡化。

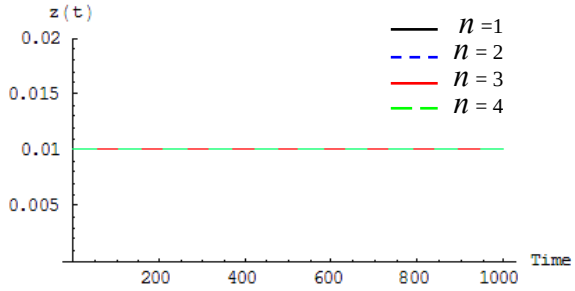

圖 5. 失效率為常數（renewal）。

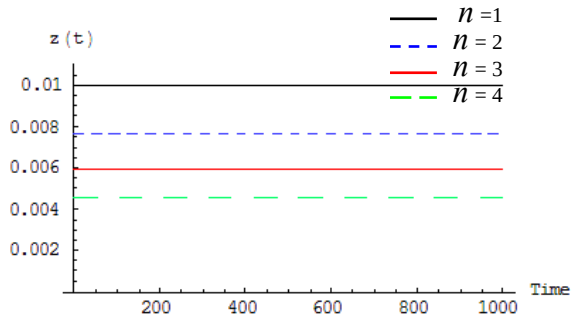

圖 6. 常數失效率型，但愈來愈好。

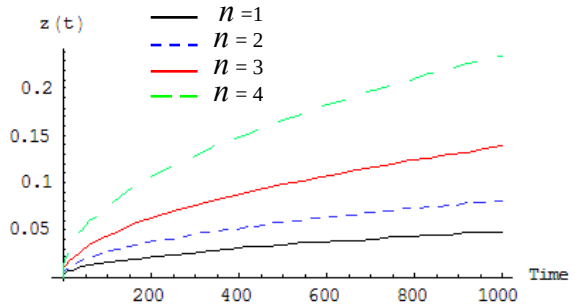

圖 7. 失效率遞增型，且逐漸惡化。

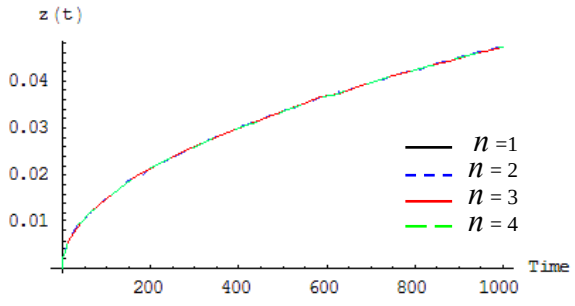

圖 8. 失效率遞增（renewal）。

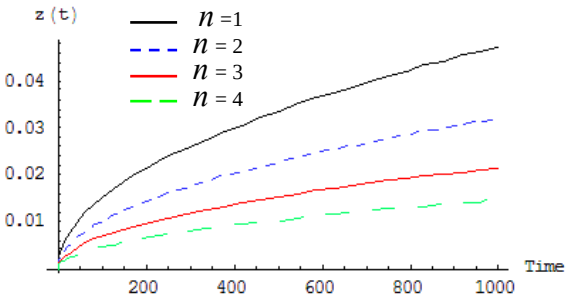

圖 9. 失效率遞增型，但愈來愈好。

因此，我們只要改變非齊性類似更新韋伯過程之可靠度模型中的參數 β 及 u，就可以一個數學式來表示軟體發生失效及更新的 9 種不同狀況，可避免使用不適當的數學模型來計算及評估軟體的可靠度。

四、模型的數值解

4.1 麥克勞林級數方法(Maclaurin Series Approach, MSA)

因為韋伯機率分配模型其 $f_{X_n}(t)$（(11)式）的 Laplace 逆運算無解析解，本研究以麥克勞林級數（Maclaurin series）近似指數函數 $e^{-(t/\eta)^{\beta}}$，以求得 $f_{X_n}(t)$ 的 Laplace 轉換之近似函數，

$$e^{-(t/\eta)^{\beta}} = \sum_{m=0}^{\infty} \frac{\left(-(t/\eta)^{\beta}\right)^m}{m!}$$

$$= 1 + \left(-(t/\eta)^{\beta}\right) + \frac{\left(-(t/\eta)^{\beta}\right)^2}{2!} + \ldots,$$

則可得

$$f_{X_n}(t) = \frac{f_{X_1}\left(u^{-(n-1)}t\right)}{u^{n-1}}$$

$$= u^{-(n-1)} \frac{\beta}{\eta}\left(\frac{u^{-(n-1)}t}{\eta}\right)^{\beta-1} e^{-\left(\frac{u^{-(n-1)}t}{\eta}\right)^{\beta}}$$

$$= u^{-(n-1)} \frac{\beta}{\eta}\left(\frac{u^{-(n-1)}t}{\eta}\right)^{\beta-1} \left(\sum_{m=0}^{\infty} \frac{\left(-\left(u^{-(n-1)}t/\eta\right)^{\beta}\right)^m}{m!} \right) \quad (13)$$

將(13)式代入(9)式可以求得類似更新函數 $Q(t)$ 的值。我們可以使用 Mathematica®或 Matlab®軟體運算，預估軟體在指定時間內之失效次數 $Q(t)$，而其中需計算(8)式的 Laplace 逆轉換，這是很複雜的數學計算問題，即使應用 Mathematica® 軟體計算，$e^{-(t/\eta)^{\beta}}$ 的麥克勞林級數近似函數以取前 10 項（m=10）較為適當，否則會造成電腦計算時間過長及記憶體不足之問題。以尺度參數(Scale Parameter) η=100，形狀參數(Shape Parameter) β=1.5 及可靠度成長參數(Reliability Growth Parameter) u=1.2 之非齊性類似更新韋伯過程為例，當麥克勞林級數取前 10 項時，分別取摺積數 2 至 10 次(n=2~10)所得到的失效次數期望值 $Q(t)$ 比較如圖 10；由於函數曲線數值相近不易觀察，將時間 t=360 時之 $Q(t)$ 值及電腦計算時間詳列如表 1（CPU 為 Intel Core2, 166GHz）。由表 1 可知，當 n 值增加時，電腦所需計算時間約以 3 的等比級數增加，當 n=10 時所需時間為 4284.973 秒（約 71 分鐘），為 n=2 時（2.184 秒）之 1962 倍；而當 n=6 與 n=7 之 $Q(t)$ 值已相當接近，僅相差 0.000373，故本案例取摺積至第 7 次即可使計算誤差至 10^{-3}。所以一般而言，取麥克勞林級數至第 10 項已可以保證計算誤差及計算時間均在可接受範圍。

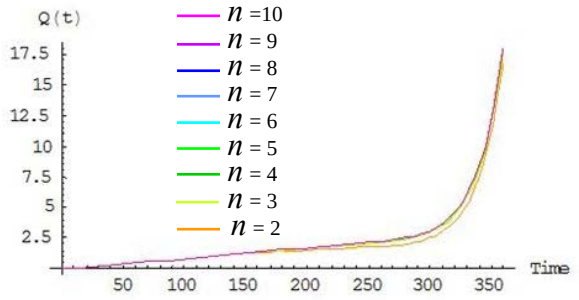

圖 10. 取麥克勞林級數至前 10 項之 $Q(t)$ 函數比較（n=2~10）。

麥克勞林級數方法計算步驟如下：

步驟 1:

因為 η、β 及 u 均為已知常數，為利於電腦計算，取摺積次數 n=10，則當 $F_{X_n}(t)$ 趨近於 1 但不大於 1 時（例如 $F_{X_n}(t)$=0.95），表示失效累積機率已接近極限，可求得一有限時間 t。

步驟 2:

將 t 帶入(9)式可以求得更新函數 $Q(t)$ 的值，當第 n 次摺積的更新次數 $Q_n(t)$ 與第 n+1 次摺積的更新次數 $Q_{n+1}(t)$ 相當接近時，則第 n+1 次摺積以後之 $Q(t)$ 增加值甚小，可以忽略

不計。此外，需注意在某一時間點 $p\,(0<p<t)$ 以後通常會發生 $Q(t)$ 值突然快速增加之不合理現象，此時需將預估時間點訂為 p，以避免數值發散。

步驟 3a：($u\geq 1$)

將上述步驟所得到的 p 及 n 值帶入更新函數 $Q(t)$，則可估算出在時間 p 內之失效次數 $Q(t)$，亦就是在時間 p 內軟體將失效 $Q(t)$ 次。需注意此時 $Q(t)$ 值應該隨 n 值增加而逐漸趨於定值(函數圖形凹口向下)，若發生 $Q(t)$ 值發散(函數圖形凹口向上)則需將預估時間點調整為反區點(inflection point)處，以為保守之預估。

步驟 3b：($0<u<1$)

因為在類似更新函數服從韋伯機率分配，且 $0<u<1$ 的條件下，$Q(t)$ 值會有逐漸發散的特性，且不像在 $u\geq 1$ 的條件下會有反區點。此時重點在求得電腦計算時間(與摺積次數有關)與類似更新函數誤差之間的平衡，避免因為進行長時間的預估，造成電腦計算時間過長(摺積次數過大)或類似更新函數誤差過大(造成發散)，因此必須以步驟 2 之圖形判斷適當的操作時間長度，然後針對該段時間進行失效次數預測。

表 1. 取麥克勞林級數至前 10 項之 $Q(t)$ 值與電腦計算時間比較(n =2~10)

摺積數 n	t =360時失效次數 $Q(t)$	電腦計算時間(秒)
2	17.035289	2.184
3	17.728059	5.445
4	17.972972	14.851
5	18.023021	43.462
6	**18.028711**	**125.409**
7	**18.029084**	**343.670**
8	18.029099	836.150
9	18.029099	1891.247
10	18.029099	4284.973

4.2 Smith and Leadbetter 方法(Smith and Leadbetter Approach, SLA)

由於我們以麥克勞林級數近似第 n 次失效的機率密度函數 $f_{X_n}(t)$ 時，類似更新函數 $Q_n(t)$ 會逐漸發散，在軟體系統愈來愈好($u\geq 1$)的狀況下，我們可以應用 SLA 方法[51]改善這個現象；但是在軟體系統逐漸惡化($0<u<1$)的狀況下，目前還沒有可以使 $Q(t)$ 值收斂的方法，故只能應用 4.1 節步驟 3b 之方法估計操作時間及失效次數。SLA 方法說明如下：

首先進行時間尺度變換

$$F(t)=1-e^{-\left(\frac{t}{\eta}\right)^{\beta}}=1-e^{-x^{\beta}}=F(x), \qquad (14)$$

則韋伯累積機率分配函數可以表示如下：

$$\left.\begin{aligned} F(x)&=1-e^{-x^{\beta}},x\geq 0\\ &=0, \qquad x<0\end{aligned}\right\} \qquad (15)$$

若 $z=x^{\beta}$，且在(15)式中限制 $\beta>0$，假設軟體系統經過 n 次除錯後的失效累積機率分配函數為 $F\left(u^{-n}x\right)$，其中 $F(x)$ 為韋伯分配函數且 $u\geq 1$，則軟體系統的操作壽命增加，因此我們可以將更新函數表示為

$$\begin{aligned} Q(x)&=F(x)+\int_0^x H\left(u^{-1}(x-z)\right)dF(z)\\ &=\sum_{k=1}^{\infty}\frac{(-)^{k-1}A_k x^{k\beta}}{\Gamma(k\beta+1)}. \end{aligned} \qquad (16)$$

對所有 x 值而言 $Q(x)$ 為絕對收斂，其中

$$\begin{aligned} &A_1=\gamma_1,\\ &A_2=\gamma_2-\gamma_1 A_1 u^{-\beta},\\ &A_3=\gamma_3-\gamma_1 A_2 u^{-2\beta}-\gamma_2 A_1 u^{-\beta}, \qquad (17)\\ &\cdots\cdots\cdots\cdots\cdots\cdots\\ &A_n=\gamma_n-\sum_{j=1}^{n-1}\gamma_j A_{n-j}u^{-\beta(n-j)} \end{aligned}$$

且

$$\gamma_n=\frac{\Gamma(n\beta+1)}{n!} \qquad (18)$$

五、案例說明

案例 1

假設某軟體之失效行為符合非齊性類似更新韋伯過程，且已知尺度參數(Scale Parameter) η =100，形狀參數(Shape Parameter) β =1.5 及可靠度成長參數(reliability growth parameter) u =1.2。接下來說明如何確定其失效行為模型，並預估一段時間內之失效次數。

解法(1)：以麥克勞林級數方法計算

步驟 1：

將 η=100, β=1.5, u=1.2 and n=10 帶入(10)式，得到軟體失效行為模型的 *Cdf* 圖形(如圖 11)，我們觀察圖 11 可以發現在 t>1151 時會產生切線斜率為負的不合理現象，故取至 t=1151，則可得 $F_X(t)$=0.953301，趨近於 1 但不大於 1。

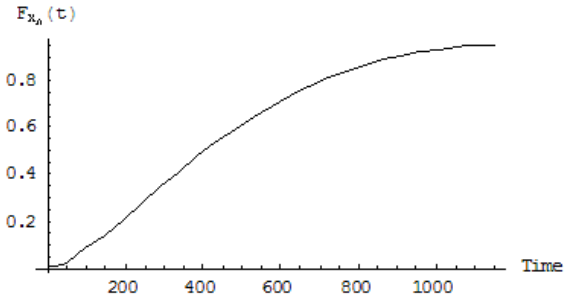

圖 11. t=1151 時，$F_{X_n}(t)$=0.953301。

步驟 2：

將已知參數 η=100, β=1.5, u=1.2 及 t=1151 帶入(13)式與(9)式，則可得到類似更新函數值 $Q(t)$，因為摺積次數 n 值愈大所需計算時間成指數比例增加，故先以 n=2 及 n=3 試算 $Q(t)$ 之值，發現 $Q(t)$ 值會因為計算誤差造成不合理之發散(如圖 12)，經試驗發現在時間 t=300 附近 $Q(t)$ 值會突然快速增加，此時需將預估時間點訂為 t=300(如圖 13)，以避免數值發散。

由圖 13 可知，類似更新函數 $Q(t)$ 在第 4 次摺積與第 5 次摺積的值已相當接近，故取 n=4 即可。

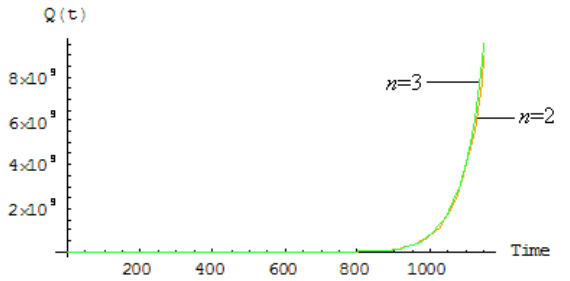

圖 12. 時間點 t=1511 時，$Q(t)$ 發散。

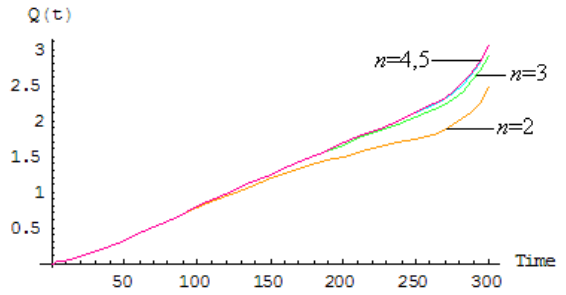

圖 13. 時間點 t=300 時，$Q(t)$ 尚未快速增加。

步驟 3：

將上述步驟所得到的 t=300 及 n=4 值帶入類似更新函數 $Q(t)$，發現 $Q(t)$ 值在時間點 249 時開始發散(函數圖形凹口向上)，故以時間點 t=249 及 n=4 再次帶入類似更新函數 $Q(t)$，估算出在 249 單位時間內之失效次數為 2.089 次，失效發生次數與時間關係如圖 14。

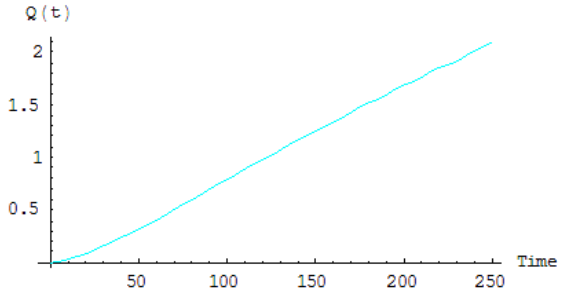

圖 14. 失效發生次數與時間關係圖。

解法(2)：以 Smith and Leadbetter 方法計算

步驟 1：

由於這個案例的軟體系統可靠度趨向於會愈來愈好(u=1.2≥1)，所以除了 MSA 方法外，我們可以用 SLA 方法使 $Q(t)$ 值收斂。而因為在較長的時間欲使 $Q(t)$ 值收斂必須取較大的 n 值，如此將造成電腦計算時間過長，故取 n=15~20 較為適當。將已知參數 η=100、β=1.5 及 u=1.2 帶入(18)、(17)及(16)式，則可以得到類似更新函數 $Q(t)$ 與時間的關係圖，由圖形的收斂狀況，我們可以判斷較適當的預估時間大約在 t=360(如圖 15)。

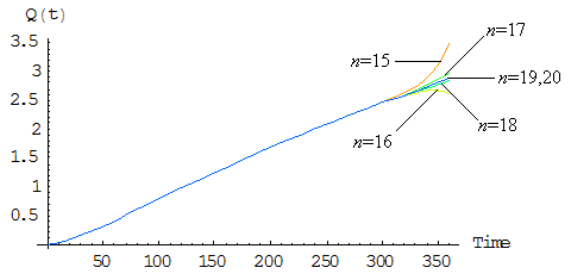

圖 15. 當 $u \geq 1$ 時，類似更新函數 $Q(t)$ 為絕對收斂。

步驟 2：

當 t =353 時，上述類似更新函數 $Q(t)$ 以 n =20 與 n =19 計算得到的誤差小於 0.01。因此取 t =353，則可以預估出軟體系統在時間內將會有 2.81864 次失效。失效次數與時間關係如圖 16。

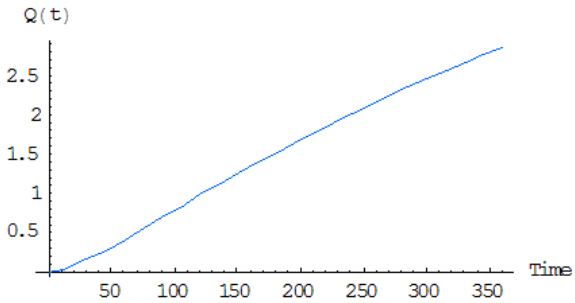

圖 16. 失效數與時間關係。

MSA 與 SLA 兩解法比較

由圖 17 我們可以發現，以 SLA 方法預估軟體系統失效，可計算至 t =353 時其誤差值小於 0.01；而以 MSA 方法，只能計算至 t =249，之後則誤差會迅速增大。因此在軟體系統可靠度趨向於會愈來愈好($u \geq 1$)的狀況下，應用 SLA 方法可以有效改善類似更新函數 $Q(t)$ 會隨預估時間而逐漸發散的現象，並增長可預估時間。

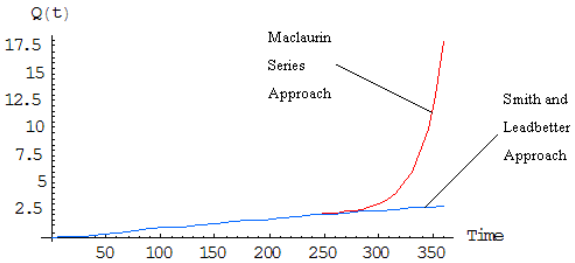

圖 17. 以 MSA 與 SLA 方法預估失效次數比較圖。

案例 2

假設某軟體之失效行為符合非齊性類似更新韋伯過程，且已知尺度參數(Scale Parameter) η =100，形狀參數(Shape Parameter) β =1.5 及可靠度成長參數(reliability growth parameter) u =0.8。那麼我們如何確定其失效行為模型，並預估一段時間內之失效次數？

解法：

由於這個案例的軟體系統可靠度為逐漸惡化的狀況($0 < u < 1$)，所以 $Q(t)$ 值無法收斂，故只能應用 4.1 節步驟 3b 之 MSA 方法估計操作時間及失效次數。我們必須在電腦計算時間與類似更新函數誤差之間求得平衡，避免因為進行長時間的預估，造成電腦計算時間過長（摺積次數過大）或類似更新函數誤差過大（造成發散）。將已知參數 η =100、β =1.5 及 u =0.8 帶入(13)、(7)、(8)及(9)式，則可以得到類似更新函數 $Q(t)$ 與時間的關係圖，分別取摺積次數 n =2,3,4,5，測試並觀察 $Q(t)$ 圖形的誤差狀況，當 t =184 時，$Q(t)$ 值在摺積次數 n =5 與 n =4 間之誤差已小於 0.1，因此我們可以判斷適當的預估時間長度為 t =184 並計算出預估失效次數為 2.14062 次(如圖 18)。

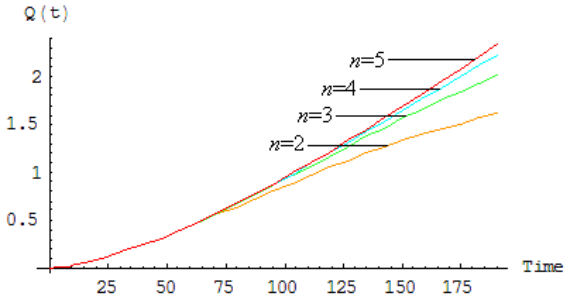

圖 18. 類似更新函數 $Q(t)$ 與時間關係圖（$0 < u < 1$，無法收斂）。

六、實際應用

本實際應用案例之軟體代號為 SS1，是一種作業系統(operating system)，由美國貝爾電話實驗室(Bell Telephone Laboratories)John D. Musa 於 1980 年 1 月公布的軟體失效資料，詳細資料來源取自於美國國防部軟體資料與分析中心(The Data & Analysis Center for Software)，網址：https://www.dacs.dtic.mil/databases/sled/swrel.php。由於案例資料註記為不完整資料，但是並未說明失效資料不完整的原因，且對於不完整資料分析方法並非本研究範圍，故本研究將案例失效資料假設為完整數據，雖然可靠度成長或惡化參數有可能與實際有所差異，但將可得到實際可靠度的低估值，對其成長或惡化之趨

勢及程度的影響不大。利用 Weibull++®軟體擬合(fit)該軟體系統最初及前二次更新共三批的資料(SS1A、SS1B 及 SS1C)，得到軟體三次失效及更新行為符合非齊性類似更新韋伯過程(擬合度 ρ 分別為 0.9922, 0.9895, 0.9921，如圖 19)，其第 1 次及第 2 次更新的可靠度成長函數分別為 $U_1(2)$ =0.900901，$U_2(3)$=0.783775。令第 3 次更新的可靠度成長函數為 $U_3(4)$，則 $U_2(3)/U_1(2)=U_3(4)/U_2(3)$，可得 $U_3(4)$=0.681187，因此我們可以由軟體系統第 2 次更新後的失效資料(SS1C) 及 $U_3(4)$ 預估第 3 次更新後失效率與時間關係，如圖 20 之綠色虛線所示，平均失效時間為 71498.24 秒。由此案例可知，本作業系統之失效率隨使用時間增加而遞減，但每次更新後卻造成惡化（如圖 1），可能因修改程式而發生新的錯誤，或是因其他軟體在此作業系統平台上執行，發生程式碼不相容、相互干擾之失效現象。若不找出原因，則下次軟體更新後將更易失效，或許軟體的功能或執行效率增加了，但卻也增加了失效風險。

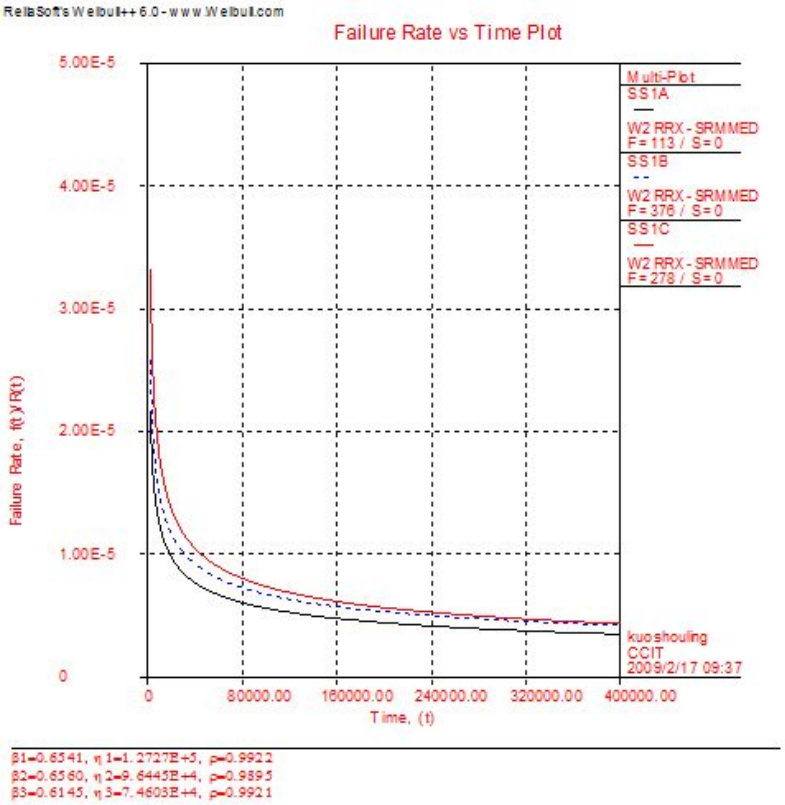

圖 19. SS1 軟體失效率與時間關係圖（非齊性類似更新韋伯過程）。

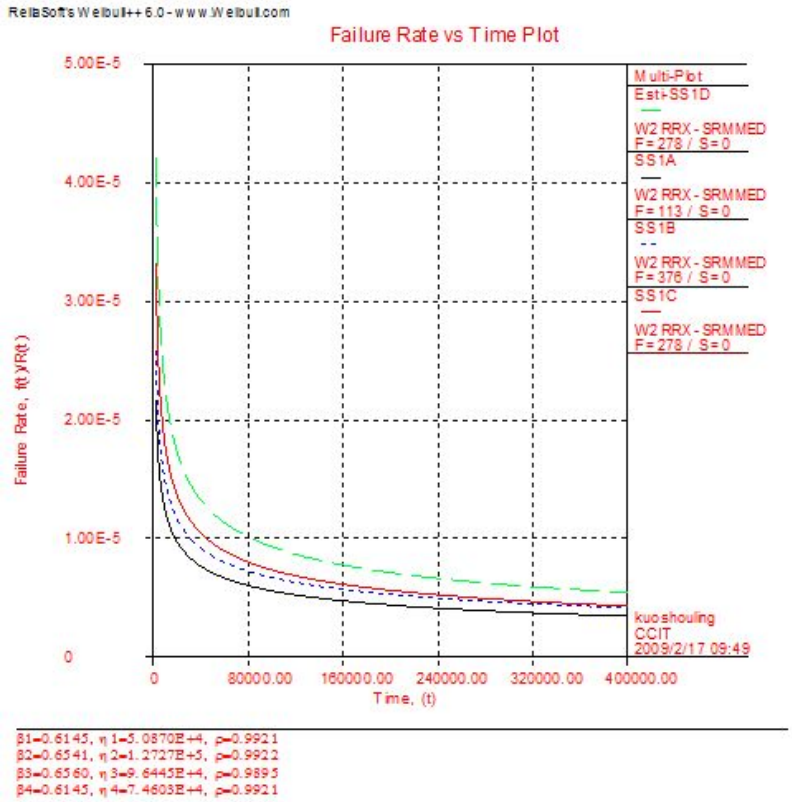

圖 20. 預估 SS1 軟體第 3 次更新後失效率與時間關係圖（綠色虛線）。

七、結果與討論

軟體可靠度模型最重要的特徵是軟體更新後的可靠度可能比原來好、和原來一樣，但也有可能比原來更差。考量以往文獻之軟體可靠度模型因無法完整描述此三個現象，因而造成在不同情況下需使用不同的軟體可靠度模型，容易讓人有無所適從之感覺。類似更新過程是以更新過程理論為基礎，更新過程最重要的假設條件為隨機變數的 iid 特性，雖然實際上 iid 特性並不常見，但是因為 iid 的假設具有容易數學量化處理的性質，為彌補此缺憾，故學者提出以類似更新過程來描述系統失效及更新行為。由於軟體可靠度具有「逐漸惡化」、「維持相同狀態」或「逐漸提昇」、等非完全更新之現象，因而可以用類似更新過程來描述，並能夠利用數學模型進行數值計算，以獲得量化分析結果。

研究的結果發現，韋伯分配函數在計算上有其困難性，在計算指數函數 $e^{-(t/\eta)^{\beta}}$ 之麥克勞林級數近似函數及 Laplace 逆轉換時，若麥克勞林級數函數項數及摺積次數過大（建議均不要超過 10），則在計算更新函數 $Q(t)$ 時因需計算複雜的 Laplace 逆轉換，易造成電腦計算時間過長及記憶體不足之問題，因而限制本研究在長時間的預估能力，僅能應用於短期內之失效預測。然而此問題在將來隨著電腦計算速度及記憶空間的增進，自然可以獲得改善。

本研究以類似更新韋伯過程建立軟體可靠度通用模型，主要考量韋伯分配函數能夠僅以單一數學函數描述軟體失效及更新的三種狀況，雖然本研究提出解決發散問題之方法及步驟目前僅能應用於短期內之失效預測，然而說明模型單一化的概念是本研究的研究重點及貢獻，解決韋伯分配函數在 Laplace 逆轉換時之發散問題則值得持續深入研究。

誌謝

本研究之進行承蒙行政院科發基金（計畫編號 NSC 96-3114-P-606-001-Y）支援研究經費，特此致謝。

參考文獻

[1] Yang, B., Li, X., Xie, M., and Tan, F., "A Generic Data-Driven Software Reliability Model with Model Mining Technique," Reliability Engineering and System Safety, Vol. 95, pp. 671-678, 2010.

[2] Lin, C. T. and Huang, C. Y., "Enhancing and Measuring the Predictive Capabilities of Testing-Effort Dependent Software Reliability Models," The Journal of Systems and Software, Vol. 81, pp. 1025-1038, 2008.

[3] Sharma, V. S. and Trivedi, K. S., "Quantifying Software Performance, Reliability and Security: An Architecture-Based Approach," The Journal of Systems and Software, Vol. 80, pp. 493-509, 2007.

[4] Choi, J. G. and Seong, P. H., "Reliability Assessment of Embedded Digital System Using Multi-State Function," Reliability Engineering and System Safety, Vol. 91, pp. 261-269, 2006.

[5] Levitin, G., "Reliability and Performance Analysis of Hardware–Software Systems with Fault-Tolerant Software Components," Reliability Engineering and System Safety, Vol. 91, pp. 570-579, 2006.

[6] Gokhale, S. S., "Software Failure Intensity, Reliability and Optimal Stopping Time Incorporating Repair Polices," International Journal of Reliability, Quality and Safety Engineering, Vol. 13, No. 5, pp. 455-477, 2006.

[7] Teng, X. and Pham, H., "A Software-Reliability Growth Model for N-Version Programming Systems," IEEE Transactions on Reliability, Vol. 51, No. 3, pp. 311-321, 2002.

[8] Huang, C. Y. and Lyu, M. R., "Optimal Testing Resource Allocation, and Sensitivity Analysis in Software Development," IEEE Transactions on Reliability, Vol. 54, No. 4, pp. 592-603, 2005.

[9] Vonta, F., Nikulin, M., Limnios, N., and Huber-Carol C., Statistical Models and Methods for Biomedical and Technical Systems, Springer-Verlag, Birkhaüser Boston, p. 138, 2008.

[10] Lyu, M. R., Handbook of Software Reliability Engineering, McGraw-Hill, New York, p. 18, 1996.

[11] Blischke, W. R. and Murthy, D. N. P., Reliability Modeling, Prediction, and Optimization, John Wiley & Son, p. 293, 2000.

[12] Pham, H., System Software Reliability, Springer-Verlag, London, p. 127, 2006.

[13] Musa, J. D., Software Reliability Engineering: More Reliable Software Faster Development and Testing, McGraw-Hill, p. 265, 1998.

[14] Musa, J. D., Software Reliability Engineering: More Reliable Software Faster Development and Testing, McGraw-Hill, p. 266, 1998.

[15] Xie, M., Software Reliability Engineering: More Reliable Software Faster Development and Testing, World Scientific Publishing Co., p. 24-26, 1991.

[16] Blischke, W. R. and Murthy, D. N. P., Reliability Modeling, Prediction, and Optimization, John Wiley & Son, p. 301, 2000.

[17] Pham, H., Handbook of Reliability Engineering, Springer-Verlag, New York, p. 214, 2003.

[18] Okamura, H., Kuroki, S., Dohi, T., and Osaki, S., "A Reliability Growth Model for Modular Software," Electronics and Communications in Japan, Part 2, Vol. 87, No. 5, pp. 43-53, 2004.

[19] Ozekici, S. and Soyer, R., "Bayesian Testing Strategies for Software with An Operational Profile," Naval Research Logistics, Vol. 48, pp. 747-763, 2001.

[20] Katerina, G. P. and Kishor, S. T., "Failure Correlation in Software Reliability Models," IEEE Transactions on Reliability, Vol. 49, No. 1, pp. 37-48, 2000.

[21] Bondavalli, A., Chiaradonna, S., Giandomenico, F. D., and Strigini, L., "A contribution to the Evaluation of the Reliability of Iterative-Execution Software," Software Testing, Verification and Reliability, Vol. 9, pp. 145-166, 1999.

[22] Baudry, B., Fleurey, F., Jezequel, J. M., and Traon, Y. L., "From Genetic to Bacteriological Algorithms for Mutation-Based Testing," Software Testing, Verification and Reliability Vol. 15, pp. 73-96, 2005.

[23] Smidts, C., Stutzke, M., and Stoddard, R. W., "Software Reliability Modeling: An Approach to Early Reliability Prediction,"

IEEE Transactions on Reliability, Vol. 47, No. 3, pp. 268-278, 1998.
[24] Weber, W., Tondok, H., and Bachmayer, M., “Enhancing Software Safety by Fault Trees- Experiences from An Application to Flight Critical Software,” Reliability Engineering and System Safety, Vol. 89, pp. 57-70, 2005.
[25] Kim, M. C. and Seong, P. H., “Reliability Graph with General Gates: An Intuitive and Practical Method for System Reliability Analysis,” Reliability Engineering and System Safety Vol. 78, pp. 239–246, 2002.
[26] Washburn, A., “A Sequential Bayesian Generalization of the Jelinski–Moranda Software Reliability Model,” Naval Research Logistics, Vol. 53, pp. 354-362, 2006.
[27] Okamura, H., Watanabe, Y., Dohi, T., and Osaki, S., “An Estimation of Software Reliability Models Based on EM Algorithm,” Electronics and Communications in Japan, Part 3, Vol. 86, No. 6, pp. 29-37, 2003.
[28] Kuo, S. Y., Huang, C. Y., and Lyu, M. R., “Framework for Modeling Software Reliability, Using Various Testing-Efforts and Fault-Detection Rates,” IEEE Transactions on Reliability, Vol. 50, No. 3, pp. 310-320, 2001.
[29] Malaiya, Y. K., Karunanithi, N., and Verma, P., “Predictability of Software-Reliability Models,” IEEE Transactions on Reliability, Vol. 41, No. 4, pp. 539-546, 1992.
[30] Bustamantea, A. S. and Bustamanteb, B. S., “Multinomial- Exponential Reliability Function: A Software Reliability Model,” Reliability Engineering and System Safety, Vol. 79, pp. 281-288, 2003.
[31] Dohi, T., Nishio, Y., Shinohara, Y., and Osaki, S. “Geometrical Solution Methods for An Optimal Software Release Problem Using Artificial Neural Networks,” Electronics and Communications in Japan, Part 3, Vol. 83, No. 8, pp. 110-118, 2000.
[32] Okamura, H., Dohi, T., and Osaki, S. “A Reliability Assessment Method for Software Products in Operational Phase—Proposal of An Accelerated Life Testing Model,” Electronics and Communications in Japan, Part 3, Vol. 84, No. 8, pp. 294-301, 2001.
[33] Tamura, Y., Yamada, S., and Kimura, M., “A Software Reliability Assessment Method Based on Neural Networks for Distributed Development Environment,” Electronics and Communications in Japan, Part 3, Vol. 86, No. 11, pp. 1236-1243, 2003.
[34] Liu, Z., Ravishanker, N., and Ray, B. K. “NHPP Models for Categorized Software Defects,” Applied Stochastic Models in Business and Industry, Vol. 21, pp. 509-524, 2005.
[35] Lawson, J., Sudweeks, J., and Scott, D., “Are New Versions of PC Operating Systems More or Less Reliable than Older Versions?,” Quality and Reliability Engineering International, Vol. 22, pp. 177-189, 2006.
[36] Levitin, G., “Optimal Structure of Fault-Tolerant Software Systems,” Reliability Engineering and System Safety, Vol. 89, pp. 286-295, 2005.
[37] Levitin, G., “Reliability and Performance Analysis for Fault-Tolerant Programs Consisting of Versions with Different Characteristics,” Reliability Engineering and System Safety, Vol. 86, pp. 75-81, 2004.
[38] Tkachtenberg, M., “A General Theory of Software-Reliability Modeling,” IEEE Transactions on Reliability, Vol. 39, No. 1, pp. 92-96, 1990.
[39] Nayak, T. K., Bose, S., and Kundu, S., “On Inconsistency of Estimators of Parameters of Non-homogeneous Poisson Process Models for Software Reliability,” Statistics and Probability Letters, Vol. 78, pp. 2217-2221, 2008.
[40]Kapur, P. K., Goswami, D. N., Bardhan, A., and Singh, O., “Flexible Software Reliability Growth Model with Testing Effort Dependent Learning Process,” Applied Mathematical Modelling, Vol. 32, pp. 1298-1307, 2008.
[41] Wang, W. L., Pan, D., and Chen, M. H., “Architecture-Based Software Reliability Modeling,” The Journal of Systems and Software, Vol. 79, pp. 132-146, 2006.
[42] Kiran, N. R. and Ravi, V., “Software Reliability Prediction by Soft Computing Techniques,” The Journal of Systems and Software, Vol. 81, pp. 576-583, 2008.
[43] Pham, H., Handbook of Reliability Engineering, Springer-Verlag, New York, p. 213, 2003.
[44] Parka, J. Y., Lee, G., and Park, J. H., “A

Class of Coverage Growth Functions and its Practical Application," Journal of the Korean Statistical Society, Vol. 37, pp. 241-247, 2008.

[45] Zhang, X. and Pham, H., "Software Field Failure Rate Prediction before Software Deployment," The Journal of Systems and Software, Vol. 79, pp. 291-300, 2006.

[46] Yang, S. C. and Kuo, S. L., "Quasi-Renewal Approach to Spares Provisioning for Deteriorating Systems under Imperfect Maintenance," 2004 Proceedings Annual Reliability and Maintainability Symposium, pp. 577-585, 2004.

[47] Ravishanker, N., Liu, Z., and Ray, B. K., "NHPP Models with Markov Switching for Software Reliability," Computational Statistics and Data Analysis, Vol. 52, pp. 3988-3999, 2008.

[48] Wang, H. Z. and Pham, H., "A Quasi-Renewal Process and its Applications in Imperfect Maintenance," International Journal of Systems Science, Vol. 27, No.10, pp. 1055-1062, 1996.

[49] Wang, H. Z. and Pham, H., "Changes to: 'A Quasi-Renewal Process and its Applications in Imperfect Maintenance'," International Journal of Systems Science, Vol. 28, No. 12, pp. 1329, 1997.

[50] Rehmert, I. J., Availability Analysis for the Quasi-Renewal Process, Ph.D. Dissertation, Virginia Polytechnic Institute and State University, Blacksburg, VA., 2000.

[51] Smith, W. L. and Leadbetter, M. R., "On the Renewal Function for the Weibull Distribution," Technometrics, Vol. 5, No. 3, pp. 393-396, 1963.