國立臺東大學資訊管理學系

碩士論文

Department of Information Science and Management Systems

National Taitung University

Master Thesis

考慮不完美除錯與錯誤生成的軟體可靠度成長模型

Software Reliability Growth Model with Imperfect Debugging and Error Generation

研究生：林哲宇

Graduate Student: Che-Yu Lin

指導教授：廖國良博士

Advisor: Gwo-Liang Liao, Ph.D.

中華民國 108 年 6 月

July, 2019

附件十

國立臺東大學
學位論文考試委員審定書

系所別：資訊管理學系碩士班

本班 林哲宇 君

所提之論文 考慮不完美除錯與錯誤生成的軟體可靠度成長模型

業經本委員會通過合於 碩士學位論文 條件

論文學位考試委員會：黃建裕

（學位考試委員會主席）

（指導教授）

論文學位考試日期：108 年 6 月 3 日

國立臺東大學

National Taitung University

國立臺東大學博、碩士學位論文授權書

本授權書所授權之論文為本人在 國立臺東大學 資訊管理學 系(所)

組 107 學年度第 2 學期取得 碩 士學位之論文。

論文名稱：考慮不完美除錯與錯誤生成的軟體可靠度成長模型

本人具有著作財產權之論文全文資料，授權予下列單位：

同意	不同意	單　位
■	□	國家圖書館
■	□	本人畢業學校圖書館
■	□	與本人畢業學校圖書館簽訂合作協議之資料庫業者

得不限地域、時間與次數以微縮、光碟或其他各種數位化方式重製後散布發行或上載網站，藉由網路傳輸，提供讀者基於個人非營利性質之線上檢索、閱覽、下載或列印。

■同意 □不同意 本人畢業學校圖書館基於學術傳播之目的，在上述範圍內得再授權第三人進行資料重製。

電子檔公開時程

立即公開	一年後公開	二年後公開	三年後公開
			✓

本論文因申請專利或發表(未申請者本條款請不予理會)需延後公開，依「教育部100年7月1日臺高(二)字第1000108377號函」，延後公開合理期限最多不超過五年。

□ 本論文已申請專利並檢附證明，專利申請案號：＿＿＿＿＿＿；

□ 本論文準備以上列論文投稿期刊。

延後公開日期：民國＿＿年＿＿月＿＿日。

上述授權內容均無須訂立讓與及授權契約書。依本授權之發行權為非專屬性發行權利。依本授權所為之收錄、重製、發行及學術研發利用均為無償。上述同意與不同意之欄位若未勾選，本人同意視同同意授權。

指導教授姓名：＿＿＿＿＿＿（親筆簽名）

研究生簽名：林哲宇（親筆正楷）

學　　號：10601302（務必填寫）

日　　期：中華民國 108 年 6 月 10 日

1. 本授權書（得自 http://portal.lib.nttu.edu.tw/lp.asp?CtNode=714&CtUnit=148&BaseDSD=52&mp=1&xq_xCat=10 下載）請以黑筆撰寫並影印裝訂於書名頁之次頁。
2. 依據91學年度第一學期第一次教務會議決議：研究生畢業論文「至少需授權學校圖書館數位化，並至遲於三年後上載網路供各界使用及校內瀏覽。」

謝誌

很快的在研究所兩年的生活即將結束了。想當初在大學時期就來到了臺東大學，四年過去了，在因緣際會下報考了臺東大學的研究所，雖然研究所並不像大學那樣有著多采多姿的活動，但是卻也讓我學習到了更多有關於專業領域上的知識。

這篇研究論文能夠順利的完成，首先要感謝指導教授廖國良老師的細心教導。每當對論文的研究遇上困難時，老師總是不厭其煩的給予幫助，使我在這兩年的期間成長了不少。也特別感謝口試委員謝昆霖老師與黃建裕老師，兩位老師在百忙之中抽空審閱我的論文，並給予不少的寶貴意見，讓我的論文不管是在內容上與結構上都能夠獲得很大的改進，使我的論文更加完整。

研究期間，感謝同班同學宏星、有合、承翰、濱壕的幫助，不管是在課業上或者是日常生活上，總在我遇到困難時，提供我意見與幫助。感謝同一實驗室的學弟仁豪與維豪，一起共同打理實驗室裡的大小事務，也在口試期間幫我們準備了許多東西。感謝多年來的好朋友，不管是共同讀研究所的、就業的，能夠在閒暇時透過通訊軟體與我聊天，一起討論對於未來的想法。最後要感謝我的家人，支持我繼續念研究所的決定，因為有你們的鼓勵，讓我能夠在遇到挫折時能夠振作起來。

很快的人生即將進入了下一個階段，相信在這幾年來的學習與歷練，將作為我面對未來的挑戰的勇氣與力量。最後，以此篇論文獻給我的家人、老師、同學以及在這段時間幫助過我的人，感謝你們！

林哲宇　謹誌

國立臺東大學資訊管理學系碩士班

中華民國一零八年六月

摘要

隨著資訊科技的發達，更多科技系統隨之應運而生。這些系統與我們的生活息息相關。一個系統能夠成功的運作是藉由軟體來幫助其運行的。對系統來說軟體是很重要的一部份，為了確保系統在使用時不會發生錯誤，軟體可靠度開始受到重視。

因此本研究提出一個基於非齊次卜瓦松過程（Non-Homogeneous Poisson Process , NHPP）的軟體可靠度成長模型（Software Reliability Growth Models, SRGMs)，考量不完美除錯及錯誤生成之情形。模型建構完成後，利用最小平方法（Least Squares Estimation, LSE）求得所需參數數值，並透過多組實際失效數據進行驗證，評估本研究提出之模型的適合度。最後使用多個模型評估標準作為模型優劣的比較基準與數個現有的軟體可靠度成長模型進行比較。

關鍵字：軟體可靠度成長模型、不完美除錯、非齊次卜瓦松過程

Abstract

More and more technological systems came into being with the development of information technology. These systems are closely related to our lives. A system can operate successfully because the software helps it to run. Software is a very important part of the systems.

In order to ensure that the system does not have errors in use, software reliability is beginning to be taken seriously. Therefore, the purpose of this study is to construct a software reliability growth model (SRGM) using Non-Homogeneous Poisson Process (NHPP) and take the possibility of error generation and imperfect debugging into consideration. After construction of the model, we use least squares estimation(LSE) to obtain the parameter values. And validation through multiple sets of actual failure data. Finally, compare to the existing model. The result shows the proposed model has better prediction ability.

Kerwords: Software reliability growth model; Imperfect debugging;
Non-Homogeneous Poisson Process.

National Taitung University

目錄

圖目錄

表目錄

第一章 緒論

第一節 研究背景與動機

隨著科技越來越發達，許多的科技系統與應用程式被廣泛的應用在食、衣、住、行、醫療還有娛樂等，這些科技系統與應用程式融入在我們的日常生活中隨處可見，與我們的生活息息相關。對於科技系統與應用程式來說，它們都是軟體所延伸出來的產物。一個科技系統或應用程式能夠有效的操作是藉由軟體來幫助其運行的。而一個軟體的大小通常也決定了軟體中程式碼的複雜度，較大型的軟體往往也有著複雜程度較高的程式碼。程式碼越是複雜也代表著軟體在運作時所發生錯誤的機率也越高，甚至是還有許多未知的錯誤隱藏在其中而尚未被發現到的，例如某一段含有錯誤的程式碼尚未被執行到，那麼這個錯誤就可能永遠不會被發現。

隨著這些大大小小的軟體產品融入我們的生活中，成為了我們生活中不可或缺的工具，我們對於這些軟體的品質也越來越重視。軟體品質對一個軟體產品來說是一個非常重要的元素之一。軟體品質的高低說明了一個軟體產品的好壞。如果一個軟體產品的品質低下，可能會導致軟體發生故障，造成了無法正常使用的情形。當上述的情況發生時，可能會使軟體使用者對軟體產品失去使用的意願，造成軟體的開發公司面臨經濟上與名聲上的損失。因此我們可以透過軟體可靠度成長模型 (Software Reliability Growth Models, SRGMs)從軟體測試收集而來數據來計算出軟體中的錯誤數量及預測軟體未來的失效行為，藉此來提升軟體產品的品質。

提升軟體品質是軟體開發人員重要的目標之一(楊睿中，2017)，開發人員在軟體開發過程中透過軟體可靠度評估進行軟體測試是不可避免的過程，這對軟體品質的控管來說是一個重要的過程。在這個階段中，軟體軟體開發人員對軟體中的錯誤進行檢測以及檢查軟體是否符合使用者的需求。為了使軟體有好的品質，軟體工程師必須盡可能的準確地去估計軟體可靠度。在近年來，隨著科技的發展、

軟體工程技術的進步，以及使用者需求提高等因素下，軟體開發者對於軟體可靠度成長模型的精確度要求也跟著提高。我們可以透過軟體可靠度成長模型來評估軟體可靠度。軟體可靠度成長模型可以預測出軟體系統中存在了多少錯誤，以及除錯後可能還剩下了多少錯誤。這種作法過去也已經被運用在實際的案例上，例如軍事武器系統、航空系統、核電廠(Kumar，2018)等。

過去有學者提出了各種軟體可靠度成長模型，目前現有的軟體可靠度成長模型中多以非齊次卜瓦松過程（Non-Homogeneous Poisson Process, NHPP）為基礎來建立的模型。以此基礎所建立的軟體可靠度成長模型是較符合實際情形的。為了讓軟體可靠度模型的預測能力更加地準確，使評估結果能夠具有更高的參考價值，因此本研究希望透過文獻的蒐集與分析，來建構一個可以有效地預測軟體錯誤數量的軟體可靠度成長模型，作為將來從事軟體開發工作的人員一個能夠參考的依據。

第二節 研究目的與範圍

過去學者所提出來的傳統軟體可靠度成長模型大多數都是以非齊次卜瓦松過程為基礎所建立出來的(Bokhair，2017)。這些傳統的軟體可靠度成長模型大多是假設軟體的失效過程是遵循某些經典的機率分佈。但是這種假設是不合理的，因為這可能會忽略了測試過程中所出現的其他隨機因素。多數的軟體可靠度成長模型假設軟體的錯誤是相互獨立的，而且檢測到的錯誤可以被完全的刪除。這一點也與現實不相符，因為某些錯誤也可能因為程式碼沒有被執行到而從未被發現。此外，這些軟體通常都是在設定好的環境下進行測試。但是實際上，這些軟體可能會在不同的環境中被不同的使用者操作(Song，2017)。所以在操作環境不確定的情況下，可能會大大的影響到了軟體的可靠度與性能。

在軟體的除錯過程中，當偵測到了錯誤時，錯誤會立即的被更正，而且並不會有新的錯誤產生，也就是完美除錯（Perfect Debugging）(林邑築，2018)。相反的，當偵測到的錯誤被更正後，可能會產生一個或多個新的錯誤，稱為不完美

除錯（Imperfect Debugging）(黃翊綺，2011)。在實際的測試過程中，可能會因為對程式碼的更動而造成了軟體中錯誤數量的增減。因此本研究以考慮不完美除錯以及錯誤生成為條件，建構出一個新的軟體可靠度成長模型。

第三節 研究方法與流程

本研究以非齊次卜瓦松過程為基礎，將不完美除錯與錯誤生成的特性加入到故障內容函數中，在固定的故障偵測率下，建構出一個新的軟體可靠度成長模型。待新模型建構完成後，運用蒐集來的實際軟體失效數據以及參數估計方法對新模型進行測試，並將測試結果利用多種模型評估標準與其他現有的軟體可靠度成長模型進行比較，藉此來觀察本研究所提出的新模型是否較其他現有的模型有更好的評估結果。

本研究之研究流程首先訂定研究目的、動機與背景。再來，根據所訂定的研究目的搜尋相關聯的軟體可靠度文獻資料，將所蒐集的資料加以整理分析出可參考的部分。模型建構部分則以非齊次卜瓦松過程為基礎，考量不完美除錯與錯誤生成的特性建構出一個新的軟體可靠度成長模型。模型建構完成後進行測試與分析。若分析結果不如預期，則進行模型的修正，重複此步驟直至模型有理想的結果。當模型確定後，與其他現有的軟體可靠度成長模型進行比較。最後就分析結果做出結論。研究流程如圖 1 所示。

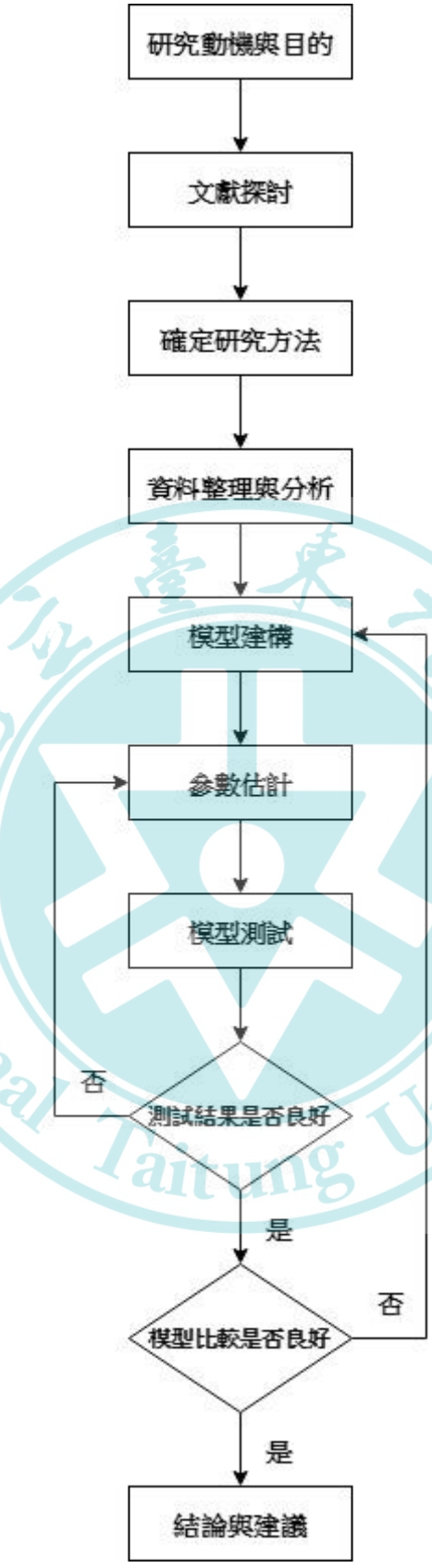
研究動機與目的
文獻探討
確定研究方法
資料整理與分析
模型建構
參數估計
模型測試
測試結果是否良好
否
是
模型比較是否良好
否
是
結論與建議

圖 1 研究流程圖

第二章 文獻探討

第一節 可靠度

可靠度（Reliability）與品質(Quality)所構成的概念類似(鍾承蓉，2018)，但是其實兩者並不相同。品質指的是在生產過程中，透過衡量成品與半成品與既有的品質規範做比較，針對其中的衡量結果做出品質管理決策。品質涵蓋的範圍非常的廣泛，從原物料、半成品、成品到顧客端等等都包含在內。主要是依照顧客所提出的需求來訂定產品品質的基準，因此品質的概念是非常注重顧客滿意度的。即產品滿足特定顧客需求的能力，顧客的需求能越被滿足則品質越高(黃大瑋，2018)。而可靠度則是有時間上的考量，通常指的是在一定的使用環境或時間條件的限制下，一個產品或是一個功能可以達到所要求的標準(翁紹仁，2011)。簡單的說，可靠度是指產品在被使用時，能夠長時間的維持預期的設計成效之能力，因此可靠度的高低也是會影響到產品或服務的滿意度的(楊邦溢，2013)。

可靠度技術的發展開始於 1950 年代，在當時美國為了解決戰場上的極高的電子裝備失效率以及過高的設備維修需求，於是於 1952 年時成立了電子裝備可靠度顧問小組(Advisory Group on Reliability of Electronic Equipment)，這是一個專門對電子設備可靠度問題進行研究的組織。該組織將可靠度定義為：「物品在既定的時間內，在特定的工作與環境條件下，執行特定性能或功能，圓滿成功達成任務的能力或機率。」(曹芷欣，2019)其他各個國家與組織也針對了可靠度做了詳細的定義，詳細整理如表 1 所示。

表 1　各個國家與組織可靠度定義

國家與組織標準	時間	可靠度定義內容
美國電子裝備可靠度顧問小組 (Advisory Group on Reliability of Electronic Equipment, AGREE)	1952	物品在既定的時間內，在特定的工作與環境條件下，執行特定性能或功能，圓滿成功達成任務的能力或機率。
歐洲品管組織 (European Organization for Quality Control, EOQC)	1965	當需要時，產品在規定的環境下成功的執行其功能一段期間的能力的量度，以機率來測量之。
美軍手冊 MIL-HDBK-217B	1970	物品在規定的使用與維護條件下，能夠執行其功能持續一段規定的變數(如時間、距離等)量度的機率。
日本國家標準 JIS Z 8115	1970	在規定條件下，物品在規定期間中能達成要求的功能之性質。
英國標準學會 (British Standard Institution, BSI)	1971	物品在規定的條件下執行需要的功能一段時間期間的機率。
英國國防標準 Def Stan00-05-1 Issue 3	1979	物品在規定的條件下執行或可以執行需要的功能持續一段規定的時間期間或操作單位的能力，此一能力可以用機率表示。
中國國家標準 CNS 11381	1985	產品於規定時間內，在所要求之環境與所給予之條件下，達到所要求機能之機率。
美國通用動力公司	1992	1. 物品在規定的條件下，可以執行期希望的功能持續一段規定的期間的機率。 2. 在規定的條件下無失效性能的機率。
大陸品質管理辭典	1992	產品在規定時間內和規定條件下，完成規定功能的能力，可靠度研究隨著科學技術發展和社會需要逐步形成一門新學科，可靠度術語也同作可靠度特性，用以表示成功的機率或成功率。

資料來源：彭鴻霖(2000)

從上述的定義來看，可以了解到可靠度所包含的要素有「條件」、「功能」、「時間」、「機率」四點(余信超，2005)。

1. 條件：確認產品在使用過程中的使用環境下可正常運作。

2. 功能：確定產品應發揮出哪些功能才算正常。

3. 時間：確定需要操作的時間。

4. 機率：將產品的可靠度用機率來表達，並且有明確的數值表示。

條件包含了環境條件、工作條件、維護條件三點，其中環境條件指的是溫度、濕度、氣壓等環境，工作條件指的是物品在工作時的動力及控制條件，如交通器械的汽油、燃料與電子設備的電壓、電流等，而維護條件包含了預防維護、定期保養、定期檢測等維護策略。

功能也可以稱為機能或性能。功能是產品開發過程中最重要的目的。每種不同的產品都有不同的功能。一個功能的正常或失效狀態需要視當初製造產品時所建立的目標或是使用者的需求而定，因此產品與產品間會有不同的功能定義。在規定產品的功能失效定義時必須描述得相當明確，以避免日後造成評估可靠度、驗證產品時，定義模糊不明確的困擾。

時間是可靠度四個要素中最重要的一點，在這裡所討論的時間指的是一段時間期間或是一段時間區間，並非指的是一個時間瞬間。許多與時間有關的參數，如操作時間、次數、距離等都包含在這裡面。每個產品的時間條件都需依照每個產品的產品生命週期來決定，因此並非所有產品都有同樣的時間條件。

機率指的是產品可靠度的整體指標。當我們使用機率來表示可靠度時，在機率與統計的理論上，這是屬於條件機率問題的範疇，其中特別強調了使用或是考慮可靠度瞬間之前及過程的條件。在同樣一件產品下，擁有不同的開始條件與不同的操作條件，所估計出來的可靠度數值也都不會完全相同。

從以上四個要素來看，可以將可靠度簡單的說明為：「產品在規定的時間與條件下，能夠成功執行一段時間的機率。」(陳銘芷，2007)，但是當我們在執行產品的操作時並不是每次都能夠成功，當我們在操作產品發生失敗造成錯誤時，這種情形我們稱之為失效(Failure)。失效指的是產品在指定的時間與條件下無法成功執行該產品的功能。失效所發生的情形可以分為三個階段，這三個階段根據失效率以及時間之間的關係可以被劃分為早夭期、穩定期、磨耗期(楊素芬，2006、

洪裕順，2018)，而這三個階段常常用浴缸曲線(Bathtub Curve)來表達，如圖 2 所示。

1. 早夭期(Infant Period)：發生於系統剛開始使用的階段，在這個階段初期失效率是很高的，但是隨著時間增加，失效率會下降，會發生這樣子的現象是因為工程師並沒有對產品進行充分的測試，或設計不良而造成產品上的缺陷，導致失效率在初期時很高的原因，但是隨著設計的改善或是系統間的磨合，失效率會隨著時間增加而開始下降。
2. 穩定期(Useful Period)：發生於系統使用的中期，在這個階段失效率處於低且穩定的狀態，成長得相當緩慢，在這個階段所發生的失效是因為產品先天設計上的限制、環境因素或使用時發生的意外所造成的。
3. 磨耗期(Wear-out Period)：發生於系統使用的後期，失效率會隨著時間的增加而快速成長，造成失效率快速成長的原因主要是因為在這個時期的系統或產品因為時間的老化或是缺乏定期維護而逐漸磨損所造成的。

圖 2 浴缸曲線

資料來源：經濟部標準檢驗局（2016）

第二節 軟體可靠度

軟體可靠度的發展得較硬體可靠度晚，其基本概念是由硬體可靠度發展而來的，軟體可靠度與硬體可靠度兩者都是屬於隨機過程，都可以藉由機率分配來進行解釋。但軟體可靠度與硬體可靠度還是有不同的地方(楊尚青、郭壽齡，2011)。兩者不同的地方在於硬體產品之所以會發生失效是因為硬體設備、零件會隨著時間的增加而逐漸的老化、磨損，因為這些老化、磨損的情形使的硬體產品在使用時發生了失效現象，最後導致可靠度降低。將硬體可靠度的失效率與時間之間的關係藉由浴缸曲線圖表達，如圖 3 所示。但是軟體產品並不會因為時間的增加而出現老化、磨損的現象。而軟體會出現失效現象的原因大多數都是由於程式碼的錯誤所造成的，由於程式碼在進行編寫時的邏輯錯誤、資料錯誤等等而產生。將軟體可靠度的失效率與時間之間的關係藉由浴缸曲線圖表達，如圖 4 所示。透過浴缸曲線圖可以發現到軟體可靠度與硬體可靠度兩者的失效率與時間之關係有很大的不同，因為軟體可以藉由軟體更新來提高軟體的功能。這裡要注意的是這裡所說的軟體更新指的是軟體功能的升級，並不是指軟體失效率的提升。在這裡我們可以看到雖然軟體產品透過了軟體的更新而提升了功能性，但是失效率還是會隨著軟體的更新而提高許多，這是因為雖然藉由軟體更新提升了軟體的功能，但是相對的也增加了軟體的複雜度，隨著軟體複雜度的提高，軟體發生失效行為的機率也會跟著提高了。

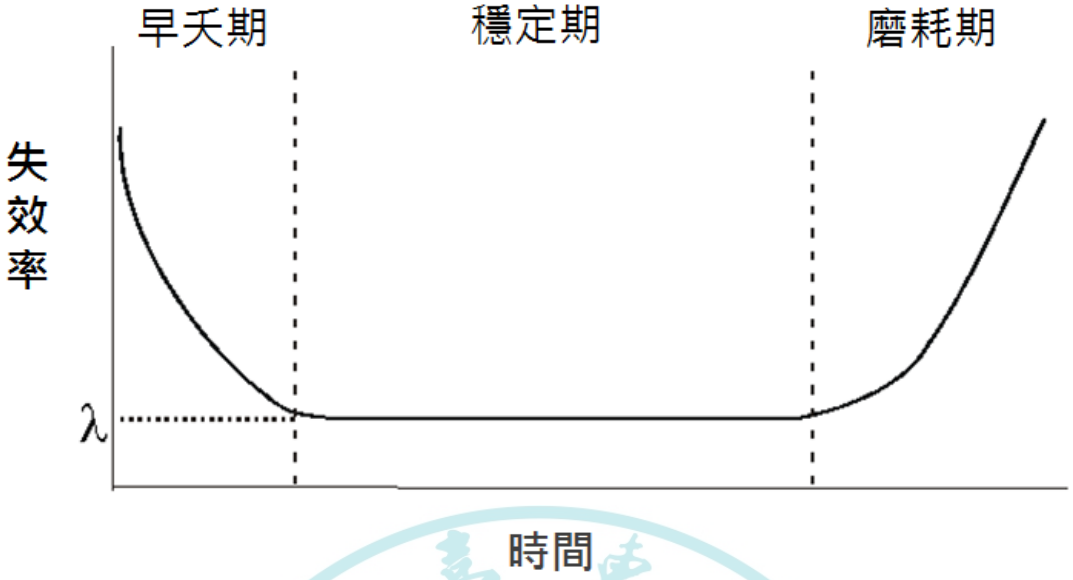

圖 3 硬體可靠度失效率與時間之關係

資料來源：Pan (1999)

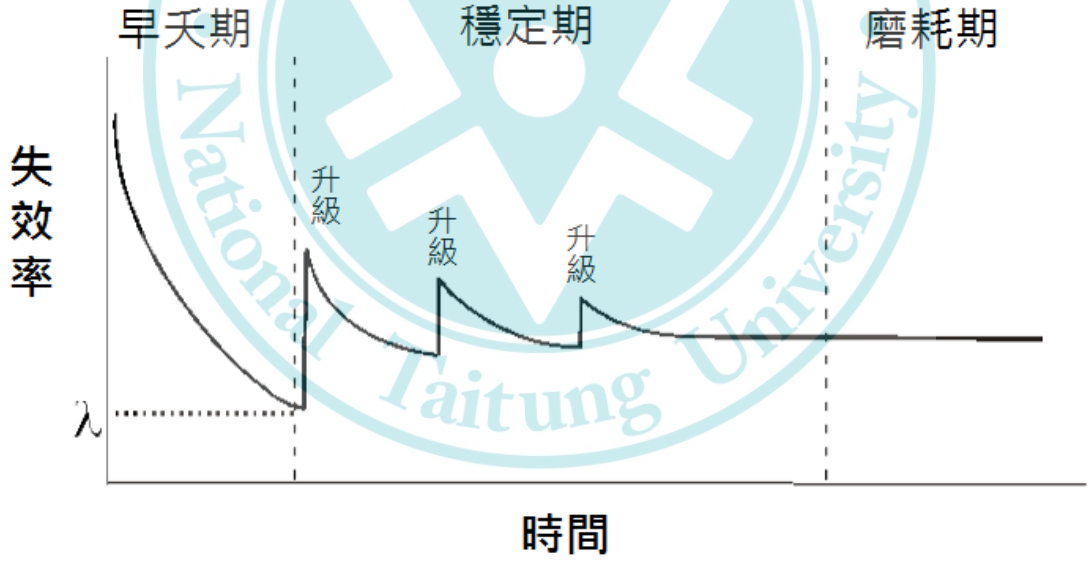

圖 4 軟體可靠度失效率與時間之關係

資料來源：Pan (1999)

軟體可靠度對於在開發中與在測試中的軟體來說是非常重要的，我們可以知道導致軟體發生失效行為的原因之一，是因為程式碼的錯誤而發生的。而且這些錯誤的數量會隨著程式碼的行數、複雜度而呈現快速的成長，如圖 5 所示。導致錯誤發生的原因可能會是因為邏輯錯誤、資料錯誤、輸入錯誤等等的原因。而且這些錯誤還必須被系統執行過才會被發現，如果在數量這麼龐大的程式碼當中，有某些部分尚未被執行，或是從來沒有被執行，那麼存在於這些程式碼當中的錯誤就永遠不會被發現到。除此之外，雖然我們將所檢測到的錯誤移除了，但是在除錯的過程中，可能會因為程式碼的更動而產生的新的錯誤。因此對整個開發與測試的過程來說是非常的複雜與耗費時間的。如果軟體開發商將帶有錯誤的軟體發佈到市場上，那麼可能會造成巨大的財物損失。但是軟體產品又不可能一直進行測試直到完全沒有錯誤了才發佈到市場上，這除了會使軟體發佈的日子遙遙無期，也會給軟體開發商帶來龐大的開發成本。如果說軟體工程師若可以找到一個最佳的停止軟體測試的時間點是非常重要的。透過剩餘的錯誤數、軟體失效率、軟體可靠度等等數據都可以用來推測最佳的停止測試時間。因此軟體可靠度評估就顯得是非常重要的了。

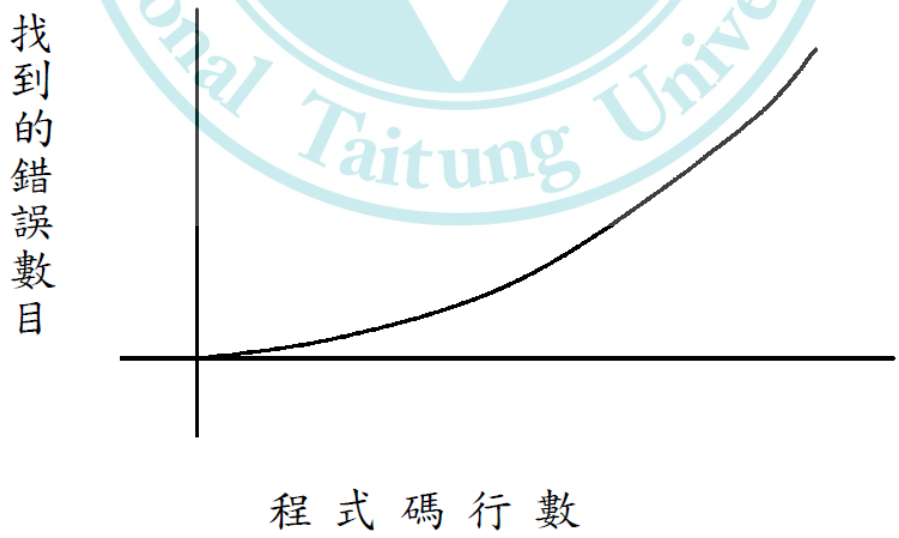

圖 5 程式碼行數與找到錯誤數之關係
資料來源：謝爾廉 (2000)

第三節 軟體除錯

Bugs 是程式中的錯誤，除錯指的是找出錯誤並消除錯誤的過程（Horton，2010）。另外，當電腦軟體執行時，發生了不正常的現象，而找出造成該現象的原因並且排除此現象的過程則稱為軟體除錯。造成 Bugs 產生的原因有很多，從基本的打字錯誤、程式碼編寫的邏輯錯誤都是可能會發生的。以下將這些原因歸類成兩種：

1. 語法錯誤：錯誤發生的情況是因為不正確的敘述所產生的，例如程式碼的結尾處少加了一個分號。
2. 語義錯誤:程式碼的語法是正確的,但是執行後的結果的並不是我們期望的。

在軟體可靠度的研究中，早期的研究大多數都將除錯過程假設為完美除錯，完美除錯指的是在除錯過程中，被偵測到的錯誤會立即的移除，而且在除錯過程中不會產生新的錯誤（Gandhi，2013）。但是這種假設是與現實不相符的，因為在實際的除錯過程中，軟體可能會因為軟體開發人員對程式碼進行了更動，而產生了一個，甚至是多個的錯誤。

不完美除錯的提出後，改善了完美除錯假設中的不足，不完美除錯指的是在除錯過程中，被偵測到的錯誤可以立即被移除並修正，但是在修正的過程中，可能會有新的錯誤產生（陳婉明，2013）。此一假設解決了軟體開發人員在對程式碼進行修正後可能會產生新錯誤的問題。

由於考慮到可能會有新的錯誤產生，因此我們還需要考量到錯誤生成率（Fault Introduction Rate）的問題，錯誤生成率指的是在修正一個錯誤時，導致產生新的錯誤的機率（Kapur，2011）。藉由將不完美除錯以及錯誤生成率兩個概念加入到軟體可靠度成長模型中，可以使我們對軟體的失效行為有更好的預測能力，也幫助軟體開發人員有更好的評估依據。

第四節 非齊次卜瓦松軟體可靠度模型

非齊次卜瓦松過程是無記憶性的，但強度函數$\lambda(t)$會隨著時間t而改變，即$\lambda(t)$為時間t之函數。當滿足下列條件時即為非齊次卜瓦松過程(朱家儀，2013)。

1. $\lambda(t)=\lim_{\Delta t\to 0}\left\{\frac{P[N(t+\Delta t)-N(t)=1]}{\Delta t}\right\}=g(t)$，其中$g(t)$為時間t之函數。
2. $P[N(t+\Delta t)-N(t)\geq 2]=o(\Delta t)$。
3. $P[N(t_0)=0]=1$。

過去幾十年來，有許多關於軟體品質與軟體可靠度的研究被提出來，而學者們也陸陸續續地提出了各種不同的軟體可靠度成長模型來估計軟體的可靠度。過去傳統的模型假設：

1. 軟體系統隨時會因為軟體的錯誤而導致軟體發生失效行為。
2. 當軟體發生失效時，造成軟體失效的錯誤會立即被移除，且不會產生新的錯誤。

而非齊次卜瓦松過程軟體可靠度成長模型則假設在時間t之前所偵測到的累積錯誤數為$N(t)$，而且$N(t)$為一個計數過程，$m(t)$為期望偵測到的錯誤數，均值函數(Mean Value Function)。均值函數$m(t)$是軟體可靠度評估中一個很重要的基本要素。在軟體可靠度評估中，我們所建立的NHPP軟體可靠度模型主要就是在尋找一個合適的均值函數$m(t)$，主要是用來表示在固定的時間t內預期會出現的錯誤數並且用來估計其機率。

NHPP軟體可靠度模型的基本假設條件為以下四點(饒國煜，2014)：

1. $N(0)=0$。
2. 有一獨立增量$\{N(t),t\geq 0\}$。
3. $P\{N(t+h)-N(t)=1\}=\lambda(t)h+o(h)$。
4. $P\{N(t+h)-N(t)\geq 2\}=o(h)$。

NHPP模型假設$N(t)$為累積錯誤數，其計數過程$\{N(t),t\geq 0\}$可以推得（江文馨，2013）：

$$\mathrm{m}(t) = \int_0^t \lambda\,(x)\,dx \tag{1}$$

以及

$$P\{N(t+x) - N(t) = k\} = \frac{[m(t+x) - m(t)]^k}{k!} e^{-[m(t+x)-m(t)]}, k = 0,1,2 \ldots \tag{2}$$

NHPP 模型假設失效強度函數 $\lambda\,(t)$與軟體內剩餘的錯誤數正比，且可以透過微分方程式求得：

$$\frac{dm(t)}{dt} = b(t)[a(t) - m(t)] \tag{3}$$

其中 $a(t)$是故障內容函數，即在時間 t 時在軟體內的總錯誤數，包含了初始錯誤及新產生錯誤，$b(t)$為在時間 t 時的故障偵測率函數，$m(t)$為均值函數，即在時間 t 時期望偵測的錯誤數。

透過給予$a(t)$跟$b(t)$不同的數值，可以求得不同的軟體可靠度模型，部分模型整理如表 2 所示，依照模型特性的不同，對於軟體可靠度評估的結果也會有所差別，一個好的軟體可靠度模型則應該具備了以下幾個特點（黃佳雄，2002）：

1. 能夠提供有效、正確的進行的預測。
2. 能夠有效的估算量化值。
3. 具有良好的基本假設。
4. 具備了廣泛的適用性。
5. 簡單。

對於軟體開發人員來說，可以利用上述五個特點與前述所說明的模型建構方法，來建立與選擇一個合適的軟體可靠度成長模型。

表 2　現有軟體可靠度模型之均值函數

模型名稱	均值函數 $m(t)$
Goel-Okumoto	$\mathrm{m}(t) = a(1 - e^{-bt})$ $\mathrm{a}(t) = a \quad \mathrm{b}(t) = b$
Delayed S-shaped	$\mathrm{m}(t) = a(1 - (1 + bt)e^{-bt})$ $\mathrm{a}(t) = a \quad \mathrm{b}(t) = (b^2 t)/(bt + 1)$
Inflection S-shaped	$\mathrm{m}(t) = a(1 - e^{-bt})/(1 + \beta e^{-bt})$ $\mathrm{a}(t) = a \quad \mathrm{b}(t) = b/(1 + \beta e^{-bt})$
HD/G-O	$\mathrm{m}(t) = log\left[(e^a - c)/\left(e^{ae^{-bt}} - c\right)\right]$ 當 $\mathrm{c} = 0$ 時與 G-O model 相同
Yamada exponential	$\mathrm{m}(t) = a\left(1 - e^{-\gamma\alpha\left(1 - e^{-\beta t}\right)}\right)$ $\mathrm{a}(t) = a$ $\mathrm{b}(t) = \gamma\alpha\beta e^{-\beta t}$
Yamada Rayleigh	$\mathrm{m}(t) = a\left(1 - e^{\gamma\alpha\left(1 - e^{-\beta t^2/2}\right)}\right)$ $\mathrm{a}(t) = a$ $\mathrm{b}(t) = \gamma\alpha\beta t e^{-\beta t^2/2}$
Yamada imperfect	$\mathrm{m}(t) = \frac{ab}{\alpha + b}(e^{\alpha t} - e^{-bt})$ $\mathrm{a}(t) = ae^{\alpha t}$ $\mathrm{b}(t) = b$
Yamada imperfect	$\mathrm{m}(t) = a(1 - e^{-bt})\left(1 - \frac{\alpha}{b}\right) + a\alpha t$ $\mathrm{a}(t) = a(1 + \alpha t)$ $\mathrm{b}(t) = b$
Pham-Zhang	$\mathrm{m}(t) =$ $\left((c + a)(1 - e^{-bt}) - \frac{ab}{b - \alpha}(e^{-\alpha t} - e^{-bt})\right)/(1 + \beta e^{-bt})$ $\mathrm{a}(t) = c + a(1 - e^{-\alpha t})$ $\mathrm{b}(t) = b/(1 + \beta e^{-bt})$
Pham-Nordmann-Zhang	$\mathrm{m}(t) = \frac{a}{1 + \beta e^{-bt}}\left[(1 - e^{-bt})\left(1 - \frac{\alpha}{b}\right) + \alpha t\right]$ $\mathrm{a}(t) = a(1 + \alpha t) \quad \mathrm{b}(t) = b/(1 + \beta e^{-bt})$

資料來源：Li (2017)

在偵測軟體錯誤的過程中，測試時間與偵測到的累積錯誤數的關係圖形被稱為軟體可靠度的成長曲線，根據曲線所呈現得不同，NHPP 模型一共被分成了指數型與 S 型兩類(Pham，2000)。

Amrit Goel 與 Kazuhira Okumoto 於 1979 年提出的 Goel-Okumoto 模型即為最初的指數型的軟體可靠度成長模型(Goel，1979)，而由 Yamada 提出的 Delayed S-shaped 模型(Yamada，1983)則為最初的 S 型軟體可靠度成長模型。

S 型軟體可靠度成長模型所描繪出的軟體可靠度成長曲線會呈現出 S 型的原因有兩點(洪翠霙，2007)：

1. 某些錯誤被其他的錯誤所隱藏，在這些錯誤被偵測或移除前，被隱藏的錯誤不會被偵測到。
2. 軟體測試還在學習階段，經驗不足的測試技巧及不熟悉的測試項目可能會導致錯誤偵測的成效不好。

透過這兩種不同類型曲線的呈現，可以解釋各種類型的軟體錯誤，作為後續用來分析軟體可靠度的參考。

第五節 學習曲線

學習曲線（Learning Curve）也可以稱為經驗曲線（Experience Curve）或改善曲線（Improvement Curve）等，如圖 6 所示，學習曲線指的是人類在進行重複性作業時，作業所需時間或成本會隨著循環次數增加而逐漸減少的現象（陳宜寧，2015），通常是用來表示單位生產時間與連續生產單位之間的關係曲線（江姮臻，2013）。透過學習曲線可以說明當一個個體或組織在執行一項作業時，透過在作業執行中學習到更多的經驗，藉此提升效率的過程。最初是由心理學發展而來的，在心理學中所定義的學習是：「因為經驗而使行為或行為潛意識產生較持久改變之歷程。」（張春興，2001），主要是在探討人類行為的基本原則與產生變化的原因（邱人文，2012）。目的是為了瞭解學習的時間或次數與結果兩者之間的關係。

相同的理論基礎也被應用在除錯過程之中。因為軟體除錯也是一項重複性的作業，並且會隨著作業次數的增減，而改變了除錯的效率。本研究將學習曲線的概念加入到軟體可靠度成長模型中，主要是因為考慮了在軟體測試期間，隨著測試時間的增加，軟體開發人員會增加對整個軟體測試的熟悉度與經驗，這有助於減少在測試過程中花費的時間，且能夠提高軟體的可靠度與除錯效率。

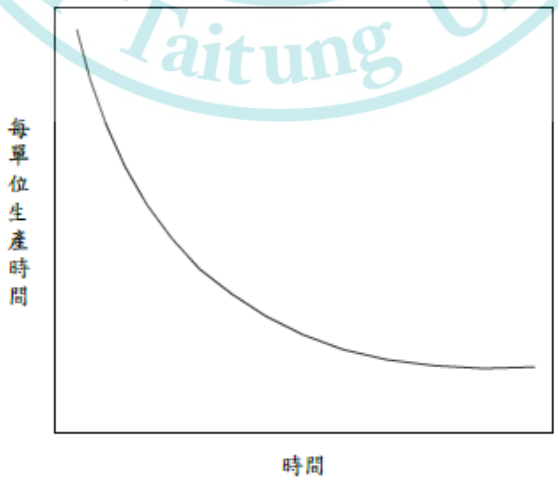

圖 6 學習曲線

資料來源：劉蕙瑜（2013）

在學習的過程中，會因為學習者的心理因素、生理因素、外在環境的干擾等等因素，使學習者所呈現出來的學習效果無法處於穩定的狀態，依照學習時期的不同與學習速度的變化，學習曲線會呈現以下幾種不同的特性（張琇惟，2008）：

1. 負加速變化：於學習期間初期時學習效果快速進步，但是在學習期間後期學習效果逐漸趨緩，如圖 7 所示。造成此現象的原因可能是學習者在學習初期時富有濃厚的興趣或是學習者已經有過學習經驗，因此可以在初期時快速進步。也可能因為作業內容是由淺入深，因此在初期時學習者較容易學習，而後期則較困難。
2. 正加速變化：於學習期間初期時學習效果進步緩慢，但是在學習期間後期學習效果逐漸提升，如圖 8 所示。造成此現象的原因可能是學習者於學習初期時缺乏學習動力，或是作業內容於初期時較為困難。
3. 正負加速變化：結合了負加速變化與正加速變化兩者先快後慢與先慢後快的情形，於初期時學習效果由慢而快，呈現正加速變化，在經過一段學習後，學習效果由快而慢，呈現了負加速變化。使學習曲線呈現了 S 型曲線，如圖 9 所示。
4. 學習高原：於學習期間後期出現進步緩慢的現象，即學習曲線趨近於平坦，如圖 10 所示。學習者學習到一定程度後便不再進步，學習曲線變的平緩的情形也可以稱為平台現象。

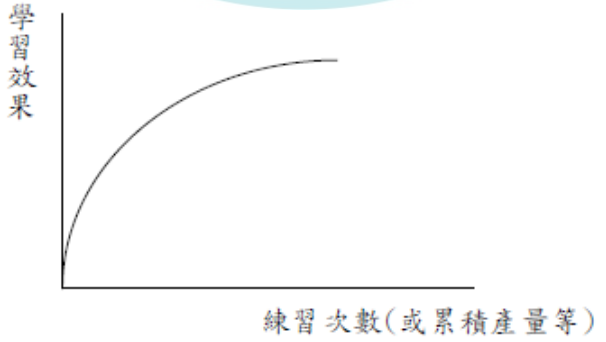

圖 7 負加速變化

資料來源：張琇惟（2008）

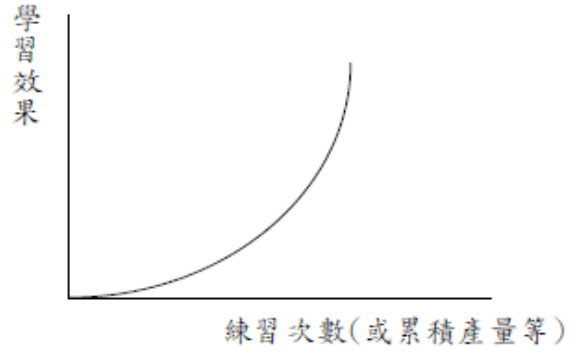

圖 8 正加速變化

資料來源：張琇惟（2008）

圖 9 正負加速變化

資料來源：張琇惟（2008）

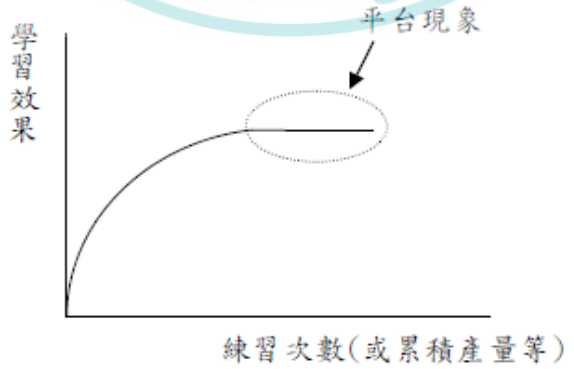

圖 10 學習高原

資料來源：張琇惟（2008）

第三章 研究方法

第一節 研究架構

本研究架構如圖 11 所示。首先藉由文獻探討了解到過去學者所提出的軟體可靠度成長模型之特性，並整理出部分資料作為本研究參考的依據。透過分析這些資料所得出的參數數值來建構模型。模型建構完成後，開始對模型進行測試。利用軟體實際失效數據與參數估計法來評估模型的參數數值，藉而取得模型與軟體實際失效數據間的適配度，了解模型的預測能力。最後，將本研究所提出的模型與現有的五個軟體可靠度成長模型進行比較。透過多種模型評估標準來判斷本研究所提出之模型是否較其他模型能夠更好的預測軟體錯誤數量。

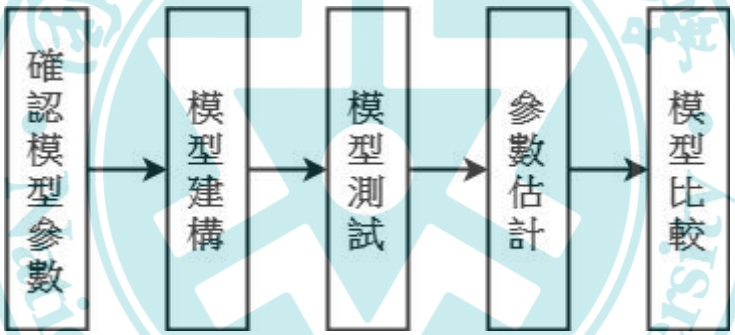

圖 11　研究架構圖

第二節 資料來源

模型建構完成後，為了有效使用並估計模型參數數值(包含本研究提出之模型與現有模型)，必須透過軟體實際失效數據進行參數估計與分析。因此，本研究將透過三組過去學者提出的文獻中蒐集而來的軟體實際失效數據集來估計各個模型的參數數值。藉此求出各個模型與軟體實際失效數據之間的適配度，用以後續進行的模型比較。三組軟體實際失效數據集如表 3 所示。

表 3　三組軟體實際失效數據集來源

數據集	軟體實際失效數據集來源	引用文獻
1	ATM 系統	Musa(1987)
2	Tandem Computers 軟體產品	Wood(1996)
3	IBM 軟體系統	Misra(1983)

第三節 參數估計法

模型建構完成後，必須將實際的軟體失效數據代入模型中，才可估計出所建構模型所需要的所有參數數值。在軟體可靠度的領域中，最常使用的參數估計方法一共有兩種，分別是最小平方法（Least Squares Estimation, LSE）與最大概似估計法（Maximum Likelihood Estimation, MLE），而本研究所採用的參數估計方法為最小平方法，以下將對最小平方法與最大概似估計法做說明。

最小平方法是使樣本觀察值與估計值的差異之平方和（Error Sum of Squares, SSE）為最小的估計法（郭又菁，2005），其公式如下：

$$\text{SSE} = \sum_{i=1}^{n} (y_i - \hat{y_i})^2 = \sum_{i=1}^{n} \left(y_i - \hat{\alpha} - \hat{\beta} x_i\right)^2 \qquad (11)$$

其中 x_i 為自變數的第 i 個觀察值，y_i 為依變數第 i 個觀察值，$\widehat{y_i}$ 為第 i 個估計值，$\hat{\alpha}$ 為迴歸模型之截距的估計值，$\hat{\beta}$ 為迴歸模型之斜率的估計值。透過將 $\hat{\alpha}$ 與 $\hat{\beta}$ 進行微分，可以得到迴歸模型的參數數值（林惠玲、陳正倉，2008）。

最大概似估計法由英國統計學家 R. A. Fisher 於 1912 年提出的估計方法，主要是假設所抽出的樣本資料，為所有可能樣本中出現機率最大的樣本。主要的應用在於一般母體之參數 θ 皆為未知，若從母體中抽出一組隨機樣本，此組樣本的可能性無法得知，因此若能找到一個估計值 $\hat{\theta}$，且可使這組樣本發生的可能性為最大時，此估計值 $\hat{\theta}$ 即為 θ 之最大概似估計值（程大器，2012）。

最大概似估計法的定義為：假設$(X_1, X_2, \ldots, X_n)$為抽自母體 f$(x;\theta)$ 之一組隨機樣本，則其概似函數（Likelihood function）即為此 n 個隨機變數$(X_1, X_2, \ldots, X_n)$的聯

合機率分配$f(X_1, X_2, \dots, X_n; \theta)$。因為參數 θ 未知，故此概似函數為 θ 的函數，一般常寫為：

$$L(\theta) = f(X_1, X_2, \dots, X_n; \theta) = f(X_1; \theta) \cdot f(X_2; \theta) \cdot \dots \cdot f(X_n; \theta) = \prod_{i=1}^{n} f(X_i; \theta) \tag{12}$$

第四節 模型評估標準

為了瞭解本研究所提出之模型是否比其他現有的軟體可靠度成長模型有較為更好的預測能力，因此必須將新模型與其他現有的軟體可靠度成長模型進行比較。軟體可靠度成長模型的性能好壞主要是透過(1)模型與軟體實際失效數據之間是否有良好的適配度。(2)模型是否能夠有效地預測軟體失效行為。為了比較這兩點，從過去到現在有學者陸陸續續地提出了不同的模型比較標準來判斷模型的優劣。

過去學者為了解決評估模型優劣的問題而提出了許多的評估標準，這些標準有均方差（Mean Square-Error, MSE）、多元判定係數（Coefficient of Multiple Determination, R^2）、相對誤差（Relative Error, RE）、乖離率（Bias Ratio, Bias）、變量（Variation）和均方根預測誤差（Root Mean Square Prediction Error, RMSPE）、隨機差異（Sum of Squares Due to Error, SSE）和 Akaike 訊息標準（Akaike Information Criterion, AIC）等等。由於模型評估標準種類眾多，本小節僅詳細介紹本研究所使用到的六個模型評估標準，作為後續比較模型時的參考，以下為模型評估標準說明及公式。

1. 均方差

均方差（Mean Square-Error, MSE），主要是透過預測值 $\hat{m}(t_k)$ 與實際觀測數據 y_i 之間的誤差作為測量標準，其公式如下（Hwang，2009）：

$$MSE=\sum_{i=1}^{n}\left(y_i - \hat{m}(t_i)\right)^2. \tag{13}$$

$\hat{m}(t_i)$ 是在時間 t_i 時，模型估計的期望累積失效數

y_i 是在時間 t_i 時，偵測到的失效數。

若 MSE 的值越低，表示適配誤差越低，代表模型的適配度越好。

2. 多元係數判定

多元係數判定（Coefficient of Multiple Determination, R^2），為了衡量解釋變數的解釋能力或迴歸方程式的適配度，將多元係數定義為可解釋的迴歸差異佔總差異的比例。其公式如下（Chiu，2008）：

$$R^2 = \frac{\sum_{i=1}^{n}(m(t_i)-\bar{y})^2}{\sum(y_i-\bar{y})^2} \quad , \quad \bar{y} = {\sum_{i=1}^{n} y_i}/{n}. \tag{14}$$

若R^2越接近 1，表示模型的適配度越好。

3. 預測比率風險

預測比率風險（Predictive-ratio Risk, PRR)，根據實際數據測量模型估計的距離與模型估計的距離。其公式如下（Pham，2003）：

$$PRR = \sum_{i=1}^{n}\left(\frac{\hat{m}(t_i) - y_i}{\hat{m}(t_i)}\right)^2. \tag{15}$$

若 PRR 的值越小，表示模型的適合度越好。

4. 預測能力

預測能力(Predictive Power, PP)，根據實際數據測量模型估算值與實際數據的距離。其公式如下（Pham，2003）：

$$PP = \sum_{i=1}^{n}\left(\frac{\hat{m}(t_i) - y_i}{y_i}\right)^2. \tag{16}$$

若 PP 的值越小，表示模型的適合度越好。

5. 變量

變量（Variation），預測誤差的標準差即為變量。其公式如下（Pillai，1997）：

$$\text{Variation} = \sqrt{\left(\frac{1}{N} - 1\right)\sum (PE_i - Bias)^2}. \tag{17}$$

其中

$$PE_i = (\hat{m}(t_i) - y_i). \tag{18}$$

$$\text{Bias} = \frac{\sum_{i=1}^{n} PE_i}{n}. \tag{19}$$

若 Variation 的值越小，代表模型的適配度越好。

6. 均方根預測誤差

均方根預測誤差（Root Mean Square Prediction Error, RMSPE）用來測量預測值跟觀察值之間的接近程度，主要透過 $Bias$ 與變量兩者加以求得。其公式如下（Pillai，1997）：

$$RMSPE = \sqrt{(Bias^2 + \text{Variation}^2)} \tag{20}$$

若 RMSPE 的值越小，代表模型的適配度越好。

第四章 模型建構

第一節 問題描述

過去幾十年來，非齊次卜瓦松過程的軟體可靠度成長模型已被許多專家學者陸續提出，但過去所提出的這些模型，大部分都是只有考慮到單一種概念而已。因此，本研究期望透過將兩種不同的概念加入到模型當中，藉由將不完美除錯與錯誤生成這兩個部分結合到故障內容函數當中，並且考慮到固定的錯誤偵測率函數，來建構出一個可使用且能夠有效預測軟體錯誤數量的軟體可靠度成長模型，作為未來軟體開發人員評估軟體可靠度的參考依據。

模型建構完成後，運用實際的軟體失效數據與現有的模型進行分析與比較，用以評估當所提出的模型運用於實際的軟體失效數據時，是否較現有的模型擁有更好的預測能力。

第二節 軟體可靠度模型建構

一、符號說明

$m(t)$	均值函數；期望在時間 t 時偵測到的錯誤數
a	軟體內初始的錯誤數
$a(t)$	故障內容函數；在時間 t 時軟體的總錯誤數
b	故障偵測率
$b(t)$	故障偵測率函數
α	錯誤生成率
β	曲折因子
r	模型組合常數

二、模型建構

所提出模型基本假設如下：

1. 錯誤的偵測與排除遵循 NHPP。
2. 由於軟體中剩餘的錯誤導致軟體在隨機時間內出現錯誤。
3. 軟體的錯誤率與軟體中剩餘的錯誤數成正比。
4. 在除錯過程中 b 是常數，且是恆定的。
5. 當軟體發生故障時，會立即刪除導致軟體故障的錯誤，且有可能產生新的錯誤。
6. 除錯過程中錯誤可能不會完全被消除，即不完美除錯。
7. a(t) 是隨著測試時間增加的指數函數。
8. 測試過程開始時的錯誤生成率高於結束時的錯誤生成率。
9. 測試團隊會隨著測試時間的增加，會越來越了解除錯過程中的知識。

根據上述的假設內容，可以獲得模型的故障內容函數與錯誤偵測率函數，兩者公式如下：

$$\mathrm{a}(t) = a(1+\alpha) + ra\left(1+e^{-\beta t}\right) \tag{21}$$

與

$$\mathrm{b}(t) = b \tag{22}$$

其中 a(t)代表了包含不完美除錯與錯誤生成的故障內容函數， b(t)代表故障偵測率。

將a(t)與b(t)帶入到方程式(3)中，並且使邊際條件m$(0) = 0$，再對方程式進行微分，可以得到所提出模型的均值函數 m(t)，公式表示如下：

$$\mathrm{m}(t) = \frac{a\cdot(e^{bt}-1) + a\cdot\alpha\cdot(e^{bt}-1) + a\cdot r\cdot(e^{bt}-1) + a\cdot b\cdot r\cdot\dfrac{e^{bt-\beta t}-1}{b-\beta}}{e^{bt}} \tag{23}$$

第三節 參數估計

參數估計的部份，本研究選擇使用最小平方估計法來為模型進行參數估計，選擇使用最小平方估計法的原因是因為該方法可以簡單的求得未知的參數估計值，且能夠使所求得的參數估計值與實際數值之間的平方和為最小。而其他用來進行比較的現有模型其參數估計方法與此方法相似，因此本研究僅列出所提出模型的參數估計過程。

假設 a、b、α、β、γ 是透過 n 組數據 $(t_0, m_0), (t_1, m_1), (t_2, m_2), \dots, (t_n, m_n)$ 所決定的，其中 m_i 為在時間 $(0, t_i)$ 內所偵測到的總錯誤數目，透過最小平方估計法可以得到模型估計函數如下：

$$\text{Min M}(a, b, \alpha, \beta, \gamma) = \sum_{i=1}^{n} \left(m_i - m(t_i)\right)^2 =$$

$$\sum_{i=1}^{n} \left(m_i - \left[(a \cdot (e^{bt} - 1) + a \cdot \alpha \cdot (e^{bt} - 1) + a \cdot r \cdot (e^{bt} - 1) + a \cdot b \cdot r \cdot (e^{(bt-\beta t)} - 1)/(b - \beta))/e^{bt}\right]\right) \tag{24}$$

透過對方程式中的 a、b、α、β、γ 進行微分，且令其偏導數(Partial Derivatives)等於零，即可得下列方程式：

$$\frac{\partial M(a, b, \alpha, \beta, \gamma)}{\partial a} =$$

$$-2 \sum_{i=1}^{n} \left[m_i - \left(a(e^{bt} - 1) + a\alpha(e^{bt} - 1) - a\gamma(e^{bt} - 1) - \left(ab\gamma(\beta t - e^{bt} + 1)/(b - \beta)\right)\right)/e^{bt}\right]$$

$$* \left[e^{bt} + \alpha(e^{bt} - 1) + \gamma(e^{bt} - 1) - \left(\frac{b\gamma(\beta t - e^{bt} + 1)}{((b - \beta) - 1)}\right)/e^{bt}\right] = 0 \tag{25}$$

$$\frac{\partial M(a,b,\alpha,\beta,\gamma)}{\partial b} =$$

$$-2a\sum_{i=1}^{n}\left[m_i-\left(a(e^{bt}-1)+a\alpha(e^{bt}-1)+a\gamma(e^{bt}-1)-\left(\frac{ab\gamma(\beta t-e^{bt}+1)}{(b-\beta)}\right)\right)/e^{bt}\right]$$

$$*\,[\beta\gamma+b^2t+\beta^2t+2b^2\gamma t+2\beta^2\gamma t-2b\beta t-\beta\gamma e^{bt}+b^2\alpha t+\alpha\beta^2t-3b\beta\gamma t-b\beta^2\gamma t^2+b^2\beta\gamma t^2$$

$$-2b\alpha\beta t]/(e^{bt}(b-\beta)^2)=0$$

(26)

$$\frac{\partial M(a,b,\alpha,\beta,\gamma)}{\partial \alpha} =$$

$$-2a\sum_{i=1}^{n}\left[m_i-\left(a(e^{bt}-1)+a\alpha(e^{bt}-1)+a\gamma(e^{bt}-1)-\left(ab\gamma(\beta t-e^{bt}+1)\right)/(b-\beta)\right)/e^{bt}\right]$$

$$*\,[e^{bt}-1]/e^{bt}=0$$

(27)

$$\frac{\partial M(a,b,\alpha,\beta,\gamma)}{\partial \beta} =$$

$$2ab\gamma\sum_{i=1}^{n}\left[m_i-\left(a(e^{bt}-1)+a\alpha(e^{bt}-1)+a\gamma(e^{bt}-1)-\left(ab\gamma(\beta t-e^{bt}+1)\right)/(b-\beta)\right)/e^{bt}\right]$$

$$*\,[bt-e^{bt}+1]/(e^{bt}(b-\beta)^2)=0$$

(28)

$$\frac{\partial M(a,b,\alpha,\beta,\gamma)}{\partial \gamma} =$$

$$-2a\sum_{i=1}^{n}\left[m_i-\left(a(e^{bt}-1)+a\alpha(e^{bt}-1)+a\gamma(e^{bt}-1)-\left(ab\gamma(\beta t-e^{bt}+1)\right)/(b-\beta)\right)/e^{bt}\right]$$

$$*\left[a(e^{bt}-1)-\left(ab(\beta t-e^{bt}+1)\right)/(b-\beta)\right]/e^{bt}=0$$

(29)

第四節 模型比較

一、軟體失效數據

1. 軟體失效數據集一

資料來源為 Musa(1987)透過即時命令和自動控制系統所收集而來的數據，數據集一在系統測試時間25個小時的CPU時間內，一共偵測到了136個錯誤數，詳細的錯誤紀錄如表4所示。

表4 即時命令和自動控制系統失效數據

測試時間(小時)	累積錯誤數	測試時間(小時)	累積錯誤數
1	27	14	111
2	43	15	116
3	54	16	122
4	64	17	122
5	75	18	127
6	83	19	128
7	84	20	129
8	89	21	131
9	92	22	132
10	93	23	134
11	97	24	135
12	104	25	136
13	106		

資料來源：Musa（1987）

2. 軟體失效數據集二

資料來源為 Wood（1996）根據 Tandem Computers 的產品所收集而來的數據，該軟體產品於測試與修正的過程中，一共發布了四個階段版本的軟體失效數

據，而本研究所採用的是第一個階段版本的軟體失效數據。第一個階段版本的系統測試時間為 20 週，測試期間內共發現 100 個錯誤，詳細的錯誤紀錄如表 5 所示。

表 5　Tandem Computers 軟體失效數據

測試時間(週)	累積錯誤數	測試時間(週)	累積錯誤數
1	16	11	81
2	24	12	86
3	27	13	90
4	33	14	93
5	41	15	96
6	49	16	98
7	54	17	99
8	58	18	100
9	69	19	100
10	75	20	100

資料來源：Wood（1996）

3. 軟體失效數據集三

資料來源為 Misra（1983）根據 IBM 的聯邦系統部門受美國 NASA 的約翰遜航空器地面處理系統中心委託所開發的軟體收集而來的數據。此數據被 Misra 認為能夠為可靠度評估提供非常準確的分析。該軟體的系統測試時間為 38 週，測試期間內所偵測到的錯誤數共有 231 個，詳細的錯誤紀錄如表 6 所示。

表 6　IBM 軟體失效數據

測試時間(週)	累積錯誤數	測試時間(週)	累積錯誤數
1	15	20	136
2	21	21	138
3	29	22	143
4	37	23	149
5	45	24	158
6	49	25	159
7	53	26	163
8	61	27	165
9	67	28	169
10	69	29	173
11	76	30	182
12	84	31	188
13	87	32	189
14	92	33	192
15	97	34	198
16	105	35	204
17	113	36	207
18	119	37	221
19	131	38	231

資料來源：Misra（1983）

二、模型測試與比較

模型建構完成之後，為了瞭解本研究所提出的模型能否在實際的環境下能夠有效的預測軟體錯誤數量，並且了解所提出的模型是否較其他現有的模型有更好的預測能力，因此將使用從實際開發出的軟體所收集到的三組軟體失效數據集來進行測試。

模型比較的部分，將選擇現有的五個軟體可靠度成長模型與所提出模型進行比較，所選擇的五個現有模型分別為 GO、DS、IS、PNZ、ROY 五個模型，如表 7 所示。以下將對各個數據集所比對分析的結果逐一進行說明。

表 7　比較模型整理

模型名稱	均值函數 $m(t)$
GO 模型	$\mathrm{m}(t) = a(1 - e^{-bt})$
DS 模型	$\mathrm{m}(t) = a(1 - (1 + bt)e^{-bt})$
IS 模型	$\mathrm{m}(t) = a(1 - e^{-bt})/(1 + \beta e^{-bt})$
PNZ 模型	$\mathrm{m}(t) = \frac{a}{1 + \beta e^{-bt}}\left[(1 - e^{-bt})\left(1 - \frac{\alpha}{b}\right) + \alpha t\right]$
ROY 模型	$\mathrm{m}(t) = a\alpha(1 - e^{-bt}) - \frac{ab}{b - \beta}\left(e^{-\beta t} - e^{-bt}\right)$
提出模型	$\mathrm{m}(t) = \frac{a \cdot (e^{bt} - 1) + a \cdot \alpha \cdot (e^{bt} - 1) + a \cdot r \cdot (e^{bt} - 1) + a \cdot b \cdot r \cdot \frac{e^{bt - \beta t} - 1}{b - \beta}}{e^{bt}}$

資料來源：本研究整理

1. 軟體失效數據集一分析結果

透過表 8 可以得知各個模型於軟體失效數據集一所估計出的參數數值，將求得的參數數值分別代入所對應模型的均值函數$\mathrm{m}(t)$中，可以估算出各個時間點所累積的錯誤數，並且能夠畫出各模型與數據集之間的適配度曲線，如圖 12 至圖

17 所示。圖 18 則描繪出了 GO、DS、IS、PNZ、提出模型與數據集一的適配度曲線。透過圖 18 可以看出本研究所提出模型描繪出的曲線是最為貼近數據集一所記錄的錯誤資料。除了透過適配度曲線的比較外，還進行了模型評估標準的比較，用以了解所提出的模型是否有較其他模型更佳的預測能力，比較結果如表 9 所示。透過表 9 可以看到本研究所提出模型在 MSE、PRR、PP、Variation 與 RMSPE 中的值皆為最小，而在 R^2 的部分則是為最接近 1 的值，在這六個模型評估標準中所提出的模型都獲得了最佳值，代表在預測軟體錯誤數上有較其他五個模型更佳的預測能力。

表 8　軟體失效數據集一各模型之參數估計結果

模型	a	b	α	β	r
GO	142.2796	0.1248	-	-	-
DS	136.9149	0.2840	-	-	-
IS	110.0752	0.2098	-	0.7141	-
PNZ	7.8692	16.0551	0.6774	8.1289	-
ROY	14.0946	0.1462	9.9870	0.0625	-
提出模型	5.4620	0.1832	4.5620	-0.0241	7.5270

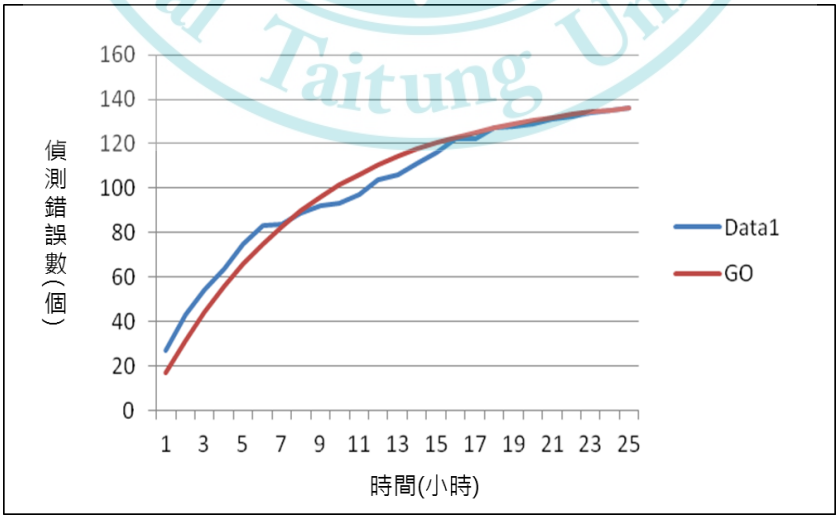

圖 12　GO 模型與數據集一之適配度曲線

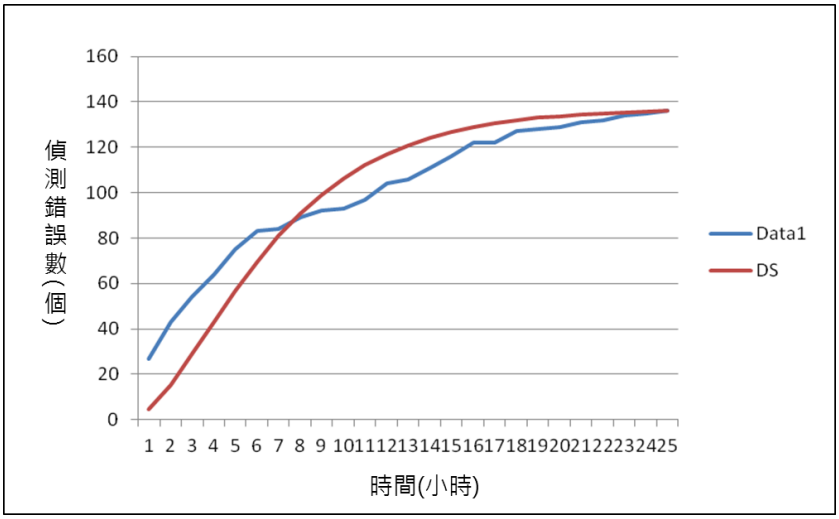

圖 13　DS 模型與數據集一之適配度曲線

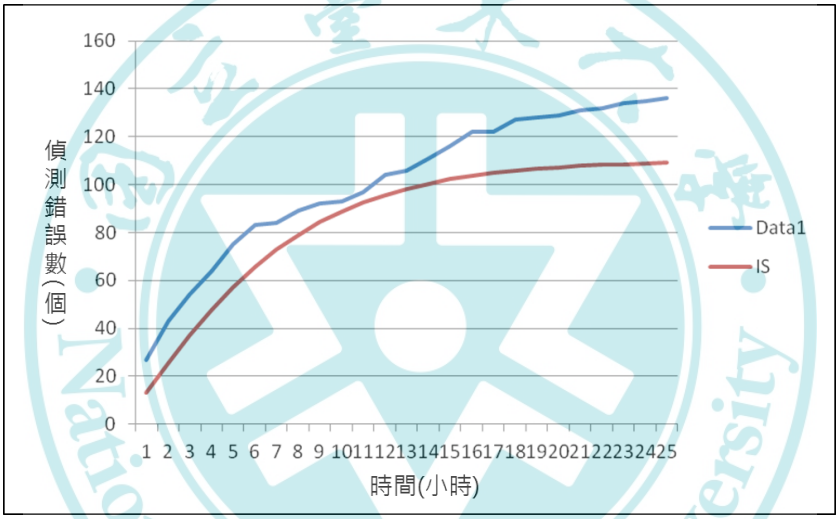

圖 14　IS 模型與數據集一之適配度曲線

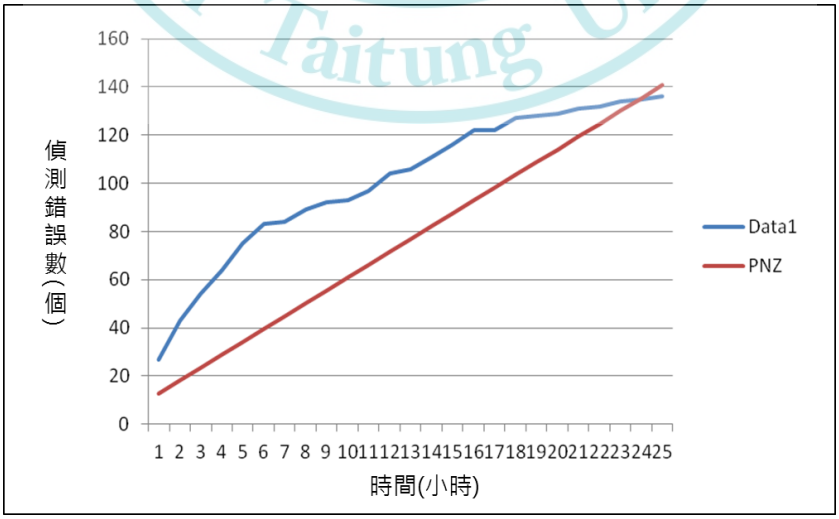

圖 15　PNZ 模型與數據集一之適配度曲線

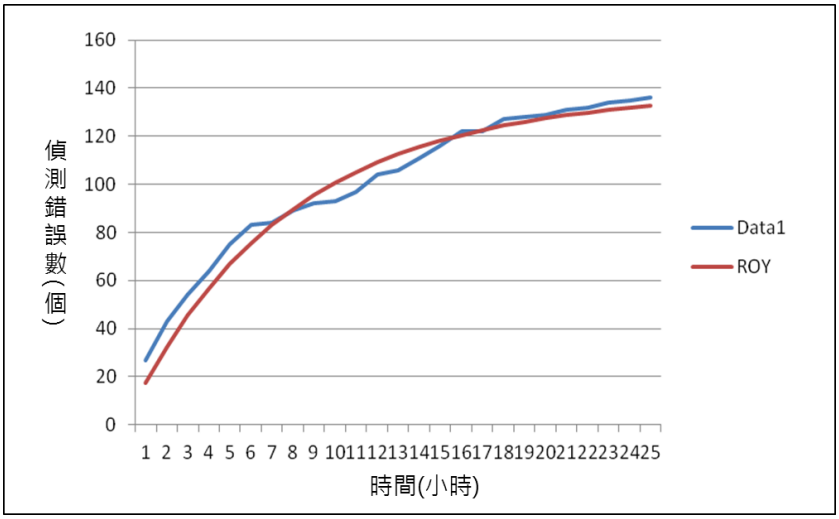

圖 16　ROY 模型與數據集一之適配度曲線

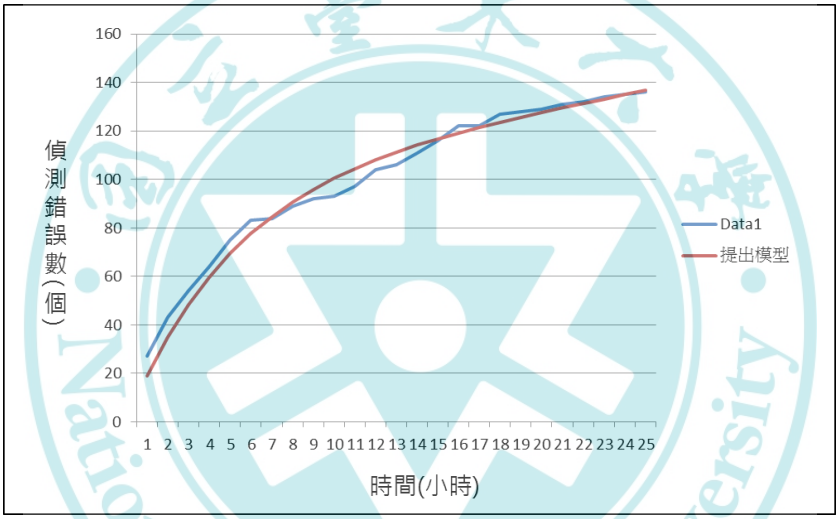

圖 17　提出模型與數據集一之適配度曲線

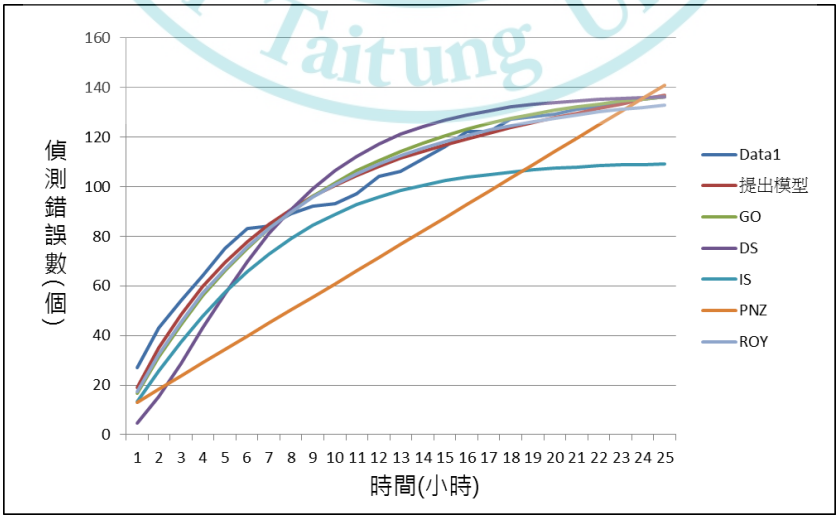

圖 18　所有模型與數據集一之適配度曲線

表 9　數據集一所有模型比較結果

模型	MSE	R^2	PRR	PP	Variation	RMSPE
GO	36.1133	0.9603	0.6443	0.3243	6.1334	6.1334
DS	168.7172	0.8148	28.5193	1.6404	13.2561	13.2569
IS	303.1079	0.6672	2.5795	1.0646	6.7501	17.4623
PNZ	768.9232	0.1558	11.9989	3.0617	13.2069	27.8550
ROY	29.6848	0.9674	0.5241	0.2751	5.3705	5.5532
提出模型	*18.3878	*0.9798	*0.2858	*0.1720	*4.3290	*4.3746

備註：*為模型評估標準中比較表現最佳者

2. 軟體失效數據集二分析結果

表 8 可以得知各個模型於軟體失效數據集二所估計出的參數數值，將求得的參數數值分別代入模型的均值函數中，可估算出各個時間點所累積的錯誤數，並且能夠畫出各模型與數據集之間的適配度曲線，如圖 19 至圖 24 所示。圖 25 則描繪出了 GO、DS、IS、PNZ、提出模型與數據集二的適配度曲線。透過圖 25 可以看出本研究所提出模型描繪出的曲線是最為貼近數據集二所記錄的錯誤資料。另外，透過模型評估標準比較，用以了解所提出的模型是否有較其他模型更佳的預測能力，比較結果如表 11 所示。透過表 11 可以得知在 MSE、R^2、PP、Variation 與 RMSPE 這五個模型評估標準中都獲得了最佳值，而在 PRR 的部分則在六個模型中排列在第三名，由 GO 模型獲得的值為最小是最佳結果。雖然在 PRR 這項標準中沒有獲得較好的結果，但在其他五項標準皆為最佳值，因此，整體來說本研究所提出模型在預測軟體錯誤數上有較其他五個模型更佳的預測能力。

表 10　軟體失效數據集二各模型之參數估計結果

模型	a	b	α	β	r
GO	110.1675	0.1194	-	-	-
DS	96.5055	0.2941	-	-	-
IS	99.8768	0.2363	-	2.1605	-
PNZ	7.0691	14.3160	0.7030	3.9385	-
ROY	11.3416	0.1054	10.2026	0.2856	-
提出模型	3.5479	0.0738	8.5133	0.0157	15.8086

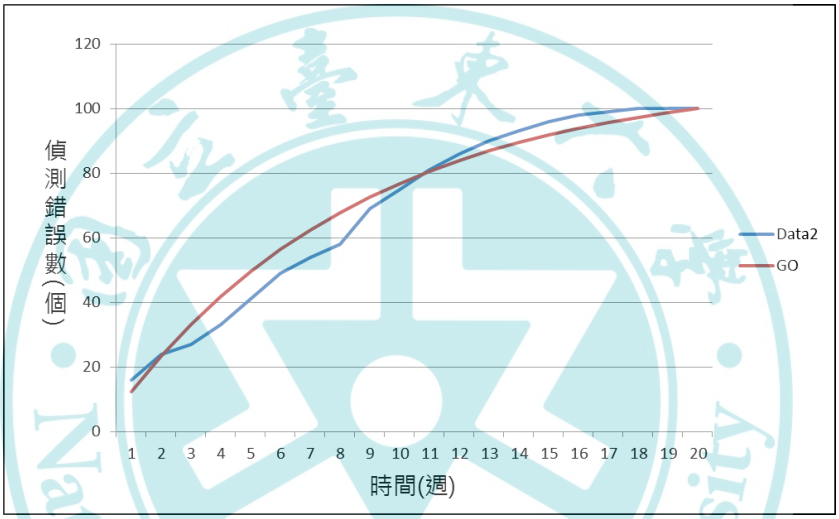

圖 19　GO 模型與數據集二之適配度曲線

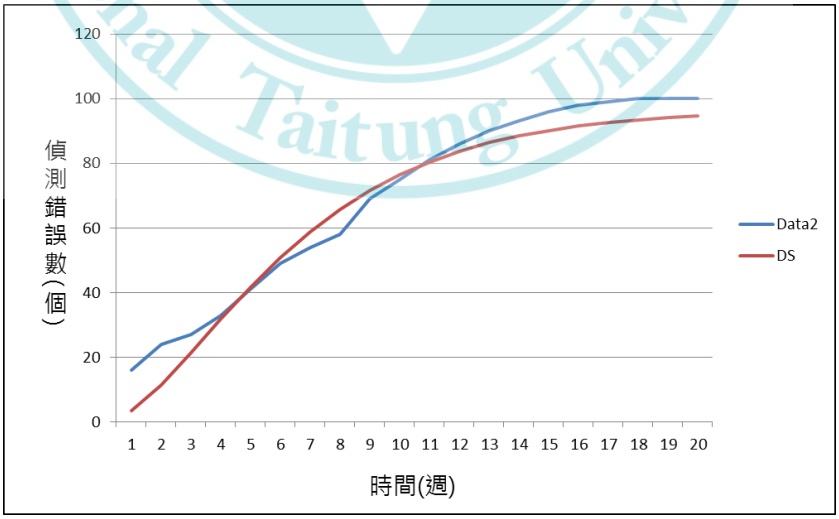

圖 20　DS 模型與數據集二之適配度曲線

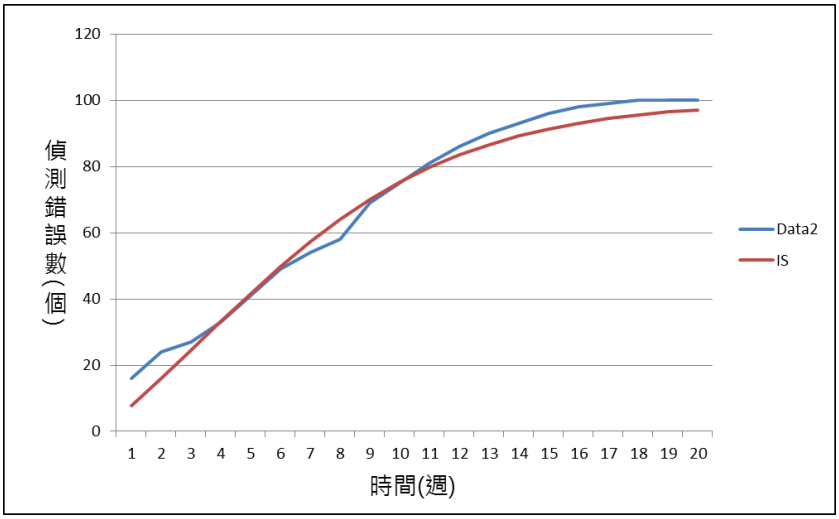

圖 21　IS 模型與數據集二之適配度曲線

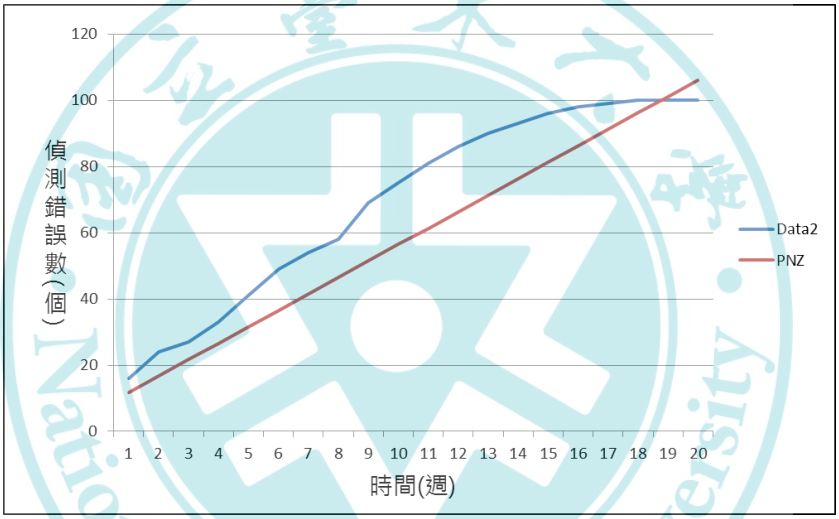

圖 22　PNZ 模型與數據集二之適配度曲線

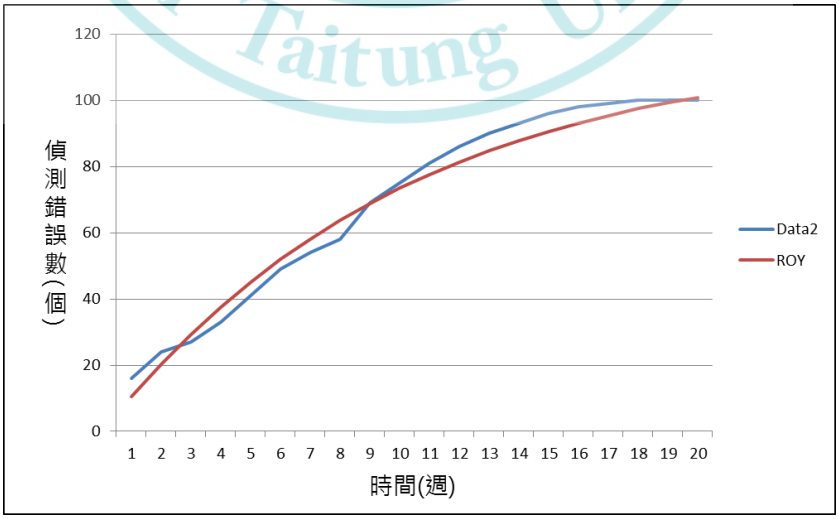

圖 23　ROY 模型與數據集二之適配度曲線

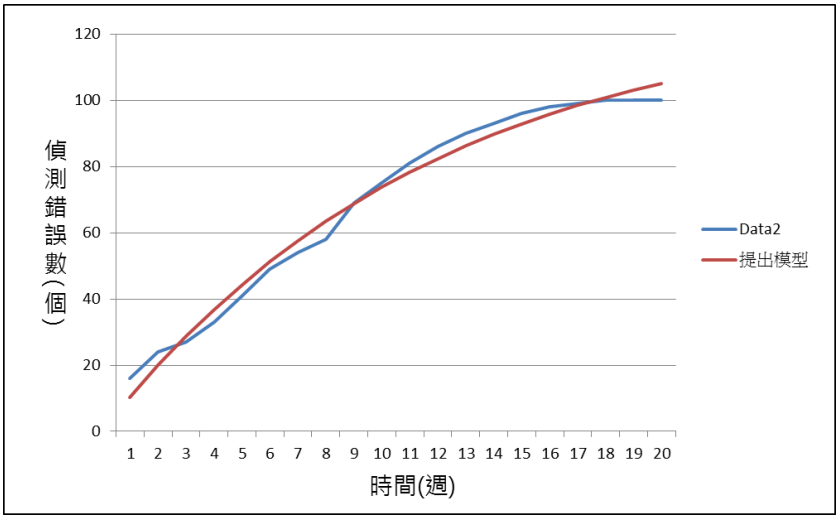

圖 24　提出模型與數據集二之適配度曲線

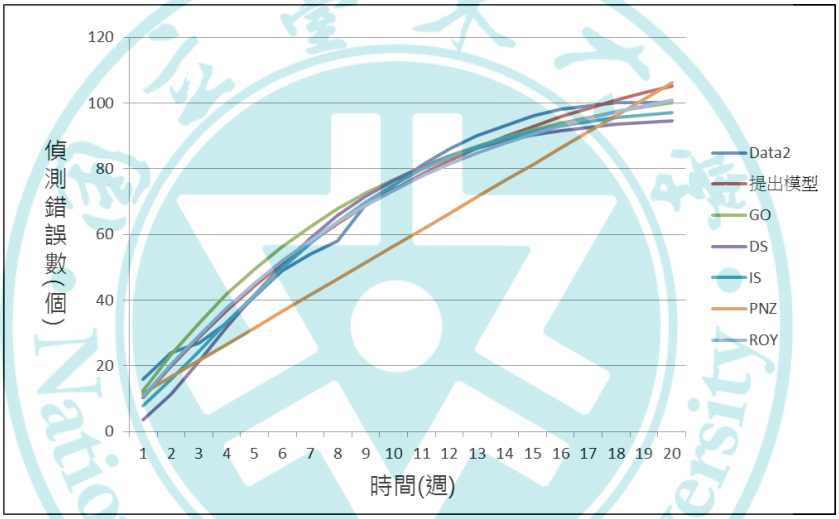

圖 25　所有模型與數據集二之適配度曲線

表 11　數據集二所有模型比較結果

模型	MSE	R^2	PRR	PP	Variation	RMSPE
GO	25.9973	0.9680	*0.2624	0.3058	5.0665	5.2091
DS	35.4303	0.9564	14.6946	0.9948	5.2588	5.9984
IS	16.4015	0.9798	1.4091	0.4131	3.5150	4.0737
PNZ	159.8183	0.8034	1.4017	0.8154	7.1687	12.3409
ROY	15.3078	0.9812	0.3595	0.2121	3.8570	3.9931
提出模型	*11.1286	*0.9863	0.3860	*0.2046	*3.4214	*3.4225

備註：*為模型評估標準中比較表現最佳者

3. 軟體失效數據集三分析結果

表 8 可以得知各個模型於軟體失效數據集三所估計出的參數數值，將求得的參數數值分別代入模型的均值函數中，可估算出各個時間點所累積的錯誤數，並且能夠畫出各模型與數據集之間的適配度曲線，如圖 26 至圖 31 所示。圖 32 則描繪出 GO、DS、IS、PNZ、提出模型與數據集三的適配度曲線。透過圖 32 可以看出本研究所提出模型描繪出的曲線是最為貼近數據集三所記錄的錯誤資料。透過模型評估標準比較結果，可以得知所提出的模型是否有較其他模型更佳的預測能力，比較結果如表 13 所示。透過表 13 可以得知在 MSE、PRR、PP、Variation 與 RMSPE 五個模型評估標準中皆為最小值，而在 R^2 的部分為最接近 1 的值。在六個評估標準中皆獲得了最佳值，因此可以得知本研究所提出模型在預測軟體錯誤數上有較其他五個模型更佳的預測能力。

表 12　軟體失效數據集三各模型之參數估計結果

模型	a	b	α	β	r
GO	185.0301	0.0425	-	-	-
DS	147.1733	0.1081	-	-	-
IS	202.6357	0.1137	-	6.0853	-
PNZ	6.0690	13.7211	0.9786	1.5102	-
ROY	23.8902	0.0142	23.2078	0.1507	-
提出模型	5.6690	0.0742	5.3253	-0.0421	8.4357

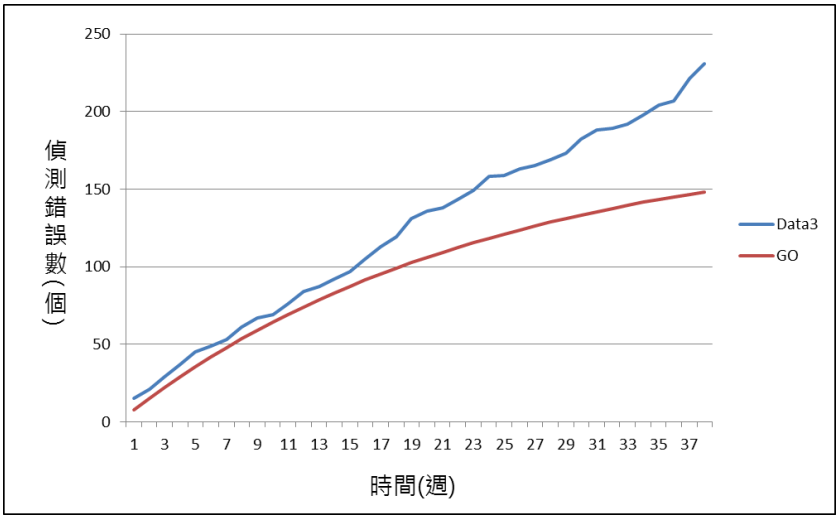

圖 26　GO 模型與數據集三之適配度曲線

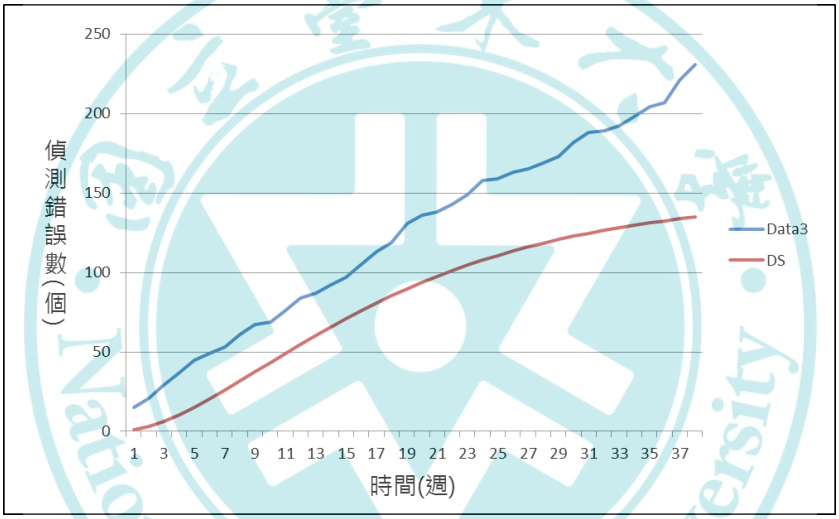

圖 27　DS 模型與數據集三之適配度曲線

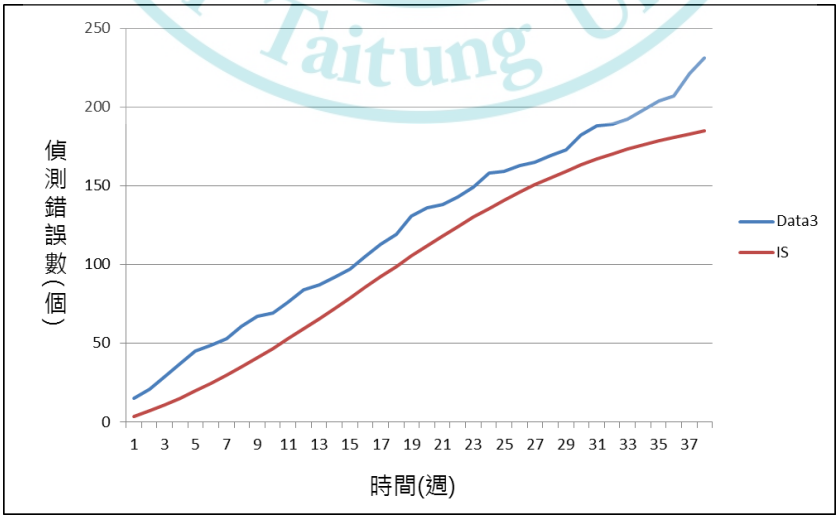

圖 28　IS 模型與數據集三之適配度曲線

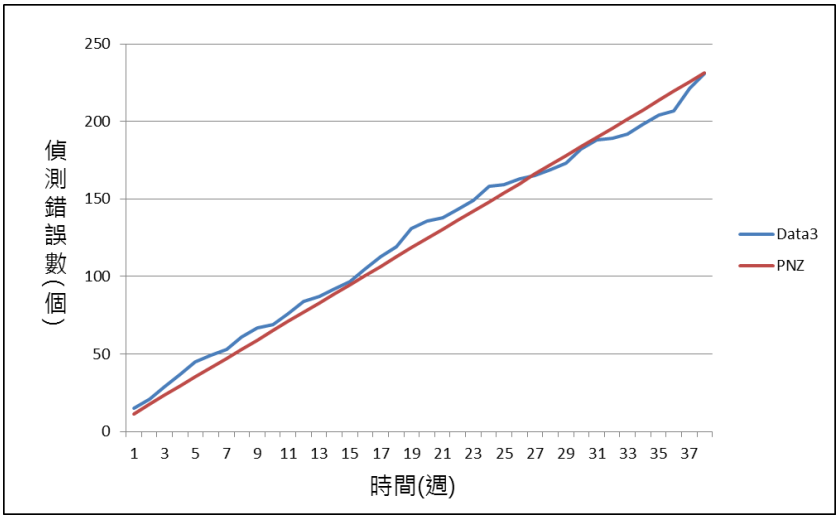

圖 29　PNZ 模型與數據集三之適配度曲線

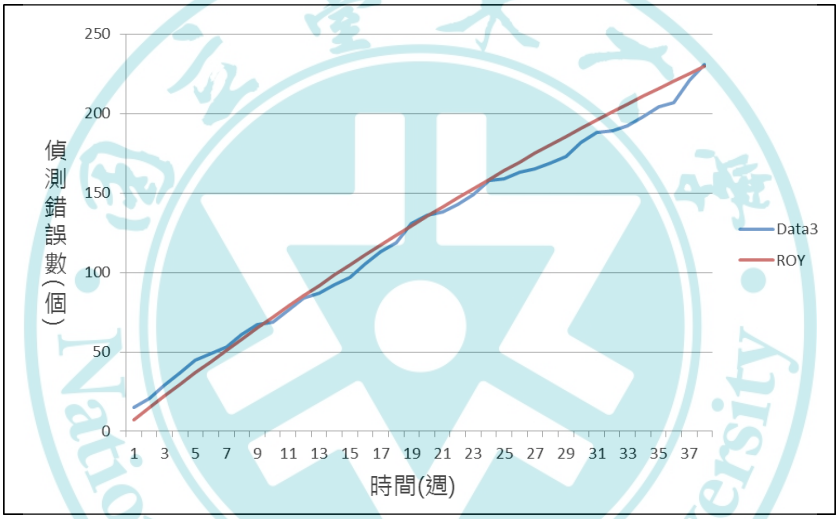

圖 30　ROY 模型與數據集三之適配度曲線

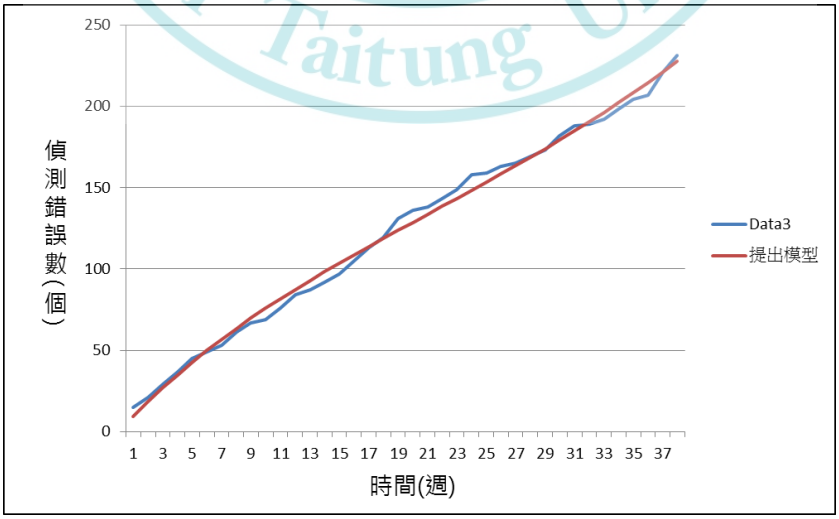

圖 31　提出模型與數據集三之適配度曲線

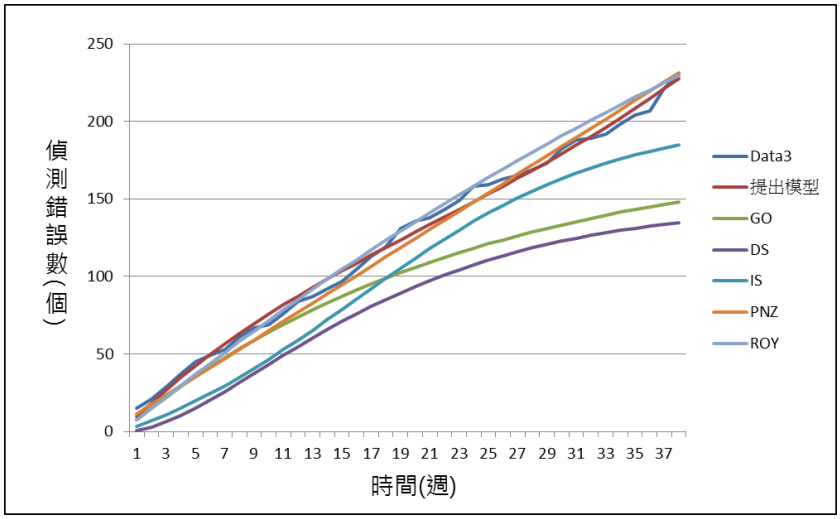

圖 32　所有模型與數據集三之適配度曲線

表 13　數據集三所有模型比較結果

模型	MSE	R^2	PRR	PP	Variation	RMSPE
GO	1303.2814	0.6490	4.2020	2.0410	21.9319	36.2759
DS	2239.1605	0.3969	386.3894	7.2690	19.6007	47.4265
IS	508.2322	0.8631	26.0648	3.7266	6.2406	22.5667
PNZ	46.0890	0.9876	0.4913	0.3412	6.3428	6.8664
ROY	51.8236	0.9860	1.4381	0.5390	6.4293	7.2740
提出模型	*20.4960	*0.9945	*0.4274	*0.2212	*4.5860	*4.5880

備註：*為模型評估標準中比較表現最佳者

第五節 小結

經由前一節的分析結果可以整理出以下幾點：

1. 本研究所提出模型在運用三組軟體實際失效數據集進行測試時，提出的模型能夠描繪出良好的適配度曲線，代表提出的模型可以有效地預測軟體錯誤數量。
2. 本研究所提出模型在運用三組軟體實際失效數據集進行測試時，透過六項模型評估標準進行評估，在兩組數據集中獲得的結果為六項模型評估標準皆為最佳，而僅在一組數據集中有一項模型評估標準排名第三，其餘皆為最佳。整體來說本研究所提出的模型在預測軟體錯誤數量上較其他模型準確。

第五章 結論與建議

第一節 結論

隨著科技的發展，軟體系統的應用也越來越普及，軟體逐漸的融入我們的日常生活當中，協助我們處理許多事物。因為軟體如此地貼近我們的生活，軟體的品質逐漸受到了重視。這些年來，專家學者陸陸續續的提出了軟體可靠度成長模型，主要的目的是用來協助軟體開發人員預測軟體錯誤數量、評估軟體的可靠度，藉此來維護軟體的品質。

因此本研究提出了一個軟體可靠度成長模型，此模型考慮了不完美除錯與錯誤生成兩個部分。在模型建構完成後，透過最小平方法估計算出模型的參數數值，然後運用了三組軟體實際失效數據對模型進行測試，繪製出適配度曲線，並使用六項模型評估標準與其他五組現有模型進行比較分析。

比較分析結果顯示，本研究所提出的模型在繪製出的適配度曲線上比其他模型更加符合實際的軟體失效數據，而在各個模型評估標準的比較上也獲得了最佳的數值，表示本研究所提出的模型在預測軟體錯誤數量方面有更加準確且有效的預測能力。

第二節 建議

在軟體開發過程中，如要發佈一個完美毫無錯誤的軟體是幾乎不可能的，因為軟體失效行為的發生是要等到包含有錯誤的程式碼被執行到才會出現，如果要讓所有的錯誤都被找到，那麼整個軟體測試的過程所耗費的時間會非常的冗長且過程複雜。但如果將測試不完善的軟體發佈到市面上，軟體發生錯誤時，除了造成使用者的不便之外，還可能對軟體開發公司造成重大的損失。因此還需考量到成本的問題，因為軟體在開發的過程是不斷地再耗費時間與金錢的，隨著軟體的開發的時間增加，所耗費的人力成本、測試成本等等也是不斷的在增加。

因此，可以建議未來的研究人員將測試時間與成本者兩個面向加入軟體可靠

度模型，思考如何在測試時間與成本之間找到一個兩者之間的平衡點，以解決前述內容所提到的成本損失問題。

參考文獻

中文部分

Horton, I（2010）。Visual C++ 2010 教學手冊。臺北市：碁峰資訊。

朱家儀（2013）。台灣地區之地震發生頻度分析-以群集卜瓦松過程為適配工具。碩士論文，東海大學統計學系，臺中市。

江妲臻（2013）。考慮學習曲線下多重產品生產線之動態人力資源配置。碩士論文，國立清華大學工業工程與工程管理學系，新竹市。

江文馨（2013）。具指數型及線性特性之不完美除錯軟體可靠度模型。碩士論文，國立臺東大學資訊管理學系，臺東縣。

余信超（2005）。六標準差設計應用於 ODM 電子產品設計品質提升之研究—以無線通訊產品為例。碩士論文，國立交通大學管理學院碩士在職專班工業工程與管理組，新竹市。

邱人文（2012）。影響學習曲線效率之人員因素探討。碩士論文，龍華科技大學商學與管理研究所，桃園市。

林邑築（2018）。多種相關失效不完美除錯模型下軟體推出時間。學士論文，國立成功大學資訊管理學系，臺南市。

林惠玲、陳正倉（2008）。基礎統計學-觀念與應用。臺北市：雙葉書廊有限公司。

洪翠霙（2007）。應用有限及無限排隊理論模型於測試與操作階段之軟體可靠度塑模。碩士論文，國立清華大學資訊工程學系，新竹市。

洪裕順（2018）。封膠環氧樹脂針對覆晶封裝產品內部孔洞之改善與可靠度分析。碩士論文，國立高雄科技大學化學工程與材料工程系碩士班，高雄市。

郭又菁（2005）。應用ROQ模式衡量及評估服務品質改善方案之財務影響-以國道客運為例。碩士論文，國立交通大學運輸科技與管理學系，新竹市。

張春興（2001）。教育心理學。臺北市：心理出版社。

張琇惟（2008）。團隊動作互動中學習曲線效果之實驗研究。碩士論文，國立中正大學企業管理研究所，嘉義縣。

陳銘芷（2007）。可靠度預估與可靠度試驗之環境因子分析－－以短距無線傳輸產品為例。朝陽學報 200709 (12 期)。臺中市。

陳宜寧（2015）。高齡對手部動作學習曲線之影響。碩士論文，國立國防大學管理學院運籌管理學系碩士班，桃園市。

陳婉明（2013）。考量錯誤分類、學習效果與轉折點之不完美除錯軟體可靠度成長模型。學士論文，國立成功大學工業與資訊管理學系，臺南市。

翁紹仁（2011）。淺談可靠度工程。品質月刊 47 卷 1 期 2011 年 1 月 44-47 頁。

曹芷欣（2019）。工具機安全系統之可靠度研究。碩士論文，國立臺灣海洋大學機械與機電工程學系，基隆市。

黃翊綺（2011）。考慮不完美除錯之軟體可靠度成長模型。學士論文，國立成功大學資訊管理學系，臺南市。

黃大瑋（2018）。顧客覺知品牌形象、產品品質對其忠誠度影響之研究-以王品集團為例。碩士論文，國立高雄師範大學成人教育研究所組織發展與領導在職進修專班，高雄市。

程大器（2012）。統計學(概要)。臺北市：高點文化。

彭鴻霖（2000）。可靠度技術手冊。中華民國品質學會，臺北市。

楊睿中(2017)。運用系統動態學預測軟體可靠度成長情形與估算軟體測試成本。碩士論文，樹德科技大學資訊管理系，高雄市。

楊邦溢（2013）。應用 RCM 與 TPM 協同規劃維修策略之績效評估。碩士論文，國立勤益科技大學工業工程與管理碩士班，臺中市。

楊素芬（2006）。品質管理。臺北市：華泰文化。

楊尚青、郭壽齡（2011）。軟體可靠度模型及失效分析。中正嶺學報第四十卷第一期第149-164頁。

經濟部標準檢驗局（2016）。加速老化試驗對眼睛防護具光學性質影響研究。

劉蕙瑜（2013）。再生能源階層成本學習曲線-以台灣風力與太陽光電為例。碩士論文，國立清華大學工業工程與工程管理學系，新竹市。

謝爾廉（2000）。具轉折點及多錯誤型態之非完美除錯軟體可靠度模型。碩士論文，淡江大學資訊管理學系，新北市。

鍾承蓉（2018）。價值創新、產品品質與品牌態度對顧客忠誠度之影響關係-以 iPhone 為例。碩士論文，嶺東科技大學企業管理系碩士班，臺中市。

饒國煜（2014）。考慮時間延遲的軟體可靠度評估的隨機模型。碩士論文，東海大學統計學系，臺中市。

英文部分

Bokhair, M.U., Siddiqui, M.A., Ahmad, N.（2017）. Testing Effort Dependent Delayed S-shaped Software Reliability Growth Model with Imperfect Debugging. International Journal on Computer Science and Engineering, 9(5), pp.138-148.

Chiu, K. C., Huang, Y. S., Lee, T. Z.（2008）. A study of software reliability growth from the perspective of learning effects. Reliability Engineering and System Safety, 93, pp.1410-1421.

Goel, A. L., Okumoto, K.（1979）.Timedependent error-detection rate model for software reliability and other performance measures. IEEE Transactions on Reliability. Vol.1, R-28, No.3, pp.206-211.

Gandhi, N., Sharma, H., Aggarwal, G., Tandon, A.（2018）. Reliability Growth Modeling for OSS: A Method Combining the Bass Model and Imperfect Debugging. Advances in Intelligent Systems and Computing, pp.23-34.

Hwang, S., Pham, H.（2009）. Quasi-renewal time-delay fault-removal consideration in software reliability modelling. IEEE Transactions on Systems, Man, and Cybernetics part A: Systems and Humans, 39(1), pp.200-209.

Kumar, P., Singh, L.K., Kumar, C.（2018）. Suitability analysis of software reliability models for its applicability on NPP systems. Quality and Reliability Engineering International, 34(8), pp.1491-1509.

Kapur, P.K., Pham, H.（2009）. A Unified Approach for Developing Software Reliability Growth Models in the Presence of Imperfect Debugging and Error Generation. IEEE Transactions on Reliability, 60(1), pp.331-340.

Li, Q., Pham, H.（2017）. NHPP software reliability model considering the uncertainty of operating environments with imperfect debugging and testing coverage. Applied Mathematical Modelling, 51, pp.68-85.

Musa, J. D., Iannino, A., Okumoto, K.（1987）.Software Reliability Measurement, Prediction, Application. McGraw-Hill.

Misra, P. N.（1983）. Software reliability analysis. IBM Syst. J, 22, pp. 262–270.

Pan, J.（1999）. Software Reliability.

Pham, H.（2000）. Software Reliability. Springer-Verlag Limited.

Pham, H.（2003）. Software reliability and cost models: perspectives, comparison, and practice. Eur. J. Oper. Res, 149 (3), pp. 475–489.

Pillai, K., Nair, V.S.S.（1997）. A Model for Software Development Effort and Cost Estimation. IEEE Transactions on Software Engineering, 23(8), pp.485-497.

Song, K.Y., Chang, I.H., Pham, H. （2017）. A Software Reliability Model with a Weibull Fault Detection Rate Function Subject to Operating Environments. Appl. Sci. 2017, 7, 983.

Wood, A.（1996）. Predicting software reliability. IEEE Comput, 29(11), pp. 69–77.

Yamada, S., Ohba, M., Osaki, S.（1983）. S-Shaped Reliability Growth Modeling for Software Error Detection. IEEE Transactions on Reliability, 32, pp.475-484.