行政院國家科學委員會專題研究計畫成果報告

以類別階層解析法測試及衡量物件導向軟體系統

Preparation of NSC Project Reports

計畫編號：NSC 87-2213-E-032-004
執行期限：86 年 8 月 1 日至 87 年 7 月 31 日

主持人：莊淇銘　　淡江大學資訊工程學系

一、中文摘要

軟體測試方法及軟體衡量法則對於提昇與保證軟體品質是個關鍵的議題。這兩者雖是屬於軟體發展周期後半階段，但在軟體工程領域中佔有其重要地位。近幾年來，物件導向程式設計概念為了解決所謂的軟體危機，而成為大家爭相採用的軟體發展技術。然而，並未考慮到針對類別階層(Class Hierarchy)來做測試與衡量評估。本計畫提出一個十分重要的因子—"單元重覆繼承"(Unit Repeated Inheritance)，來當作物件導向類別階層解析法中的基礎元素。據此因子，我們描述了"繼承階層技術"(Inheritance Level Technique)，來指引使用者分別測試類別階層之軟體錯誤與衡量類別階層之軟體複雜度。此技術可以顯示出繼承性與物件導向軟體複雜度之關係，也可以揭櫫過度使用重覆(repeated inheritance)或多重(multiple inheritance)繼承會增加軟體的複雜度與易導致軟體錯誤。同時，我們也提出兩種測試標準：在內階層優先標準(intra level first)及內部階層優先標準(inter level first)來作為測試準則。

本計畫主要之目的是提出一類別階層解析法，以"自動化"的方式來偵測及衡量類別階層之軟體錯誤與複雜度，並發展與利用一組軟體工具來建構測試準則與衡量法則。最後，這個自動化之機構可提供使用者一個良好的軟體發展程序，以及完成一個整合性的物件導向軟體環境。

關鍵詞：軟體測試，軟體衡量，物件導向思維模式，類別階層，單元重覆繼承，繼承階層技術。

Abstract

Software testing methodologies and metrics are key issues to improve and assure software quality. They are important areas in the research of software engineering. However, not many focus on the testing criteria and metrics evaluation of a class hierarchy concerned. This project introduces an important factor named URI (Unit Repeated Inheritance) to realize integrated object-oriented testing and object-oriented metrics. The approach describes an ILT (Inheritance Level Technique) method as a guide to detect object-oriented software errors and measure the complexity of a class hierarchy. The proposed technique shows that inheritance has a close relation to object-oriented software complexity and reveals that overuse of repeated (multiple) inheritance will increase software complexity and be prone to implicit software errors. Also, two test criteria intra level first and inter level first are presented.

The major purpose of this project is to develop a suite of software tools to test and measure object-oriented software in an automatic manner. The measurement of the class hierarchy metric and some testing criteria thus can be formed based on the proposed mechanism. Conclusively, the automatic mechanism provides users a good approach to following the software development process and forms a well-

integrated object-oriented software environment.

Keywords: Concurrent programs, software testing criterion, software complexity, Ada language, rendezvous.

二、緣由與目的

在資訊工業快速蓬勃發展的今天，軟體工程的概念早已被廣泛地應用於軟體發展環境中，而軟體工程的目標不外乎是希望利用較低的成本來發展出較高品質的軟體，以及提高軟體發展人員之生產力。

目前，大部份的軟體計畫均面臨預算透支，進度延遲以及低可靠度等問題，軟體發展人員無法確切掌握軟體發展過程之變動因素，以致影響軟體之品質以及生產力，因而導致近幾年來大家常聽到一個術語"軟體危機"。為了因應軟體危機已發展出不少軟體工程技術，像是電腦輔助軟體工程〔CASE〕，快速的原型製作〔Quick Prototyping〕，以及資料庫管理和所謂的第四代程式語言〔4GL〕等等。這些技術的確是將軟體發展的效率提昇了一些，但仍不足解決日益增加的軟體複雜度以及修改的問題。於是物件導向程式設計〔Object-Oriented Programming〕這種新的革命性軟體系統發展觀念乃被廣泛使用。然而，物件導向程式設計被大量運用在真實世界中，主要原因在於軟體設計者能夠產生可再用之軟體元件，利用這些元件將來可大大降低軟體開發成本及人力資源之消耗。但對於目前物件導向技術之研究和經驗，大多侷限在分析〔Analysis〕，設計〔Design〕及程式設計〔Programming〕方面之研究。以軟體品質之保證與管制必須仰賴軟體測試方法與軟體衡量法則來完成，故其重要性是不可否認的。

至今，已有許多以程序導向為基礎之軟體測試方法及軟體衡量法則被提出。雖然傳統方法有其有效性，但是仍無法不做任何調整而完全適用於物件導向系統上。因為在物件導向軟體系統中，最基本的單元是"類別"而不是"副程式"，所以軟體測試應著重在類別和物件上。再者，那是不可能獨立去測試一個類別的所有內部運算，因為這些運算彼此之間會依著呼叫方式而互相溝通。而在軟體衡量法則上，也該考慮類別階層之架構。因此，發展以物件導向程式語言為基礎之軟體測試方法與軟體衡量法則，實為刻不容緩之研究方向。

於是，在物件導向軟體測試方法及軟體衡量法則尚未成熟發展之際，希望藉由我們所提出之一套用以測試及衡量類別階層之物件導向軟體系統，以使物件導向技術所開發之軟體，能夠得到最佳之軟體品質及正確性；同時讓軟體發展生命週期更臻完善。

物件導向軟體測試及衡量法則自動化〔Automatic Object-Oriented Testing and Metrics〕：

物件導向程式設計已成為 1990 年代軟體開發的主流。然而有關物件導向軟體測試的研究依然相當稀少，更不用談到"自動化"的測試工具了。軟體測試的自動化，已經成為軟體業者或軟體工程師對於系統測試的一個主要目標。其目的就是想藉由軟體自動來產生測試案例〔Test case〕，並透過這些案例的執行來確保軟體的可靠度，並及早發現錯誤以便做更正。就根據我們對於物件導向軟體的設計與發展，發現如果能在軟體的設計階段，就能夠有自動化的工具來協助發展的話，不僅有助於及早發現錯誤以降低成本外，而且有助於發展者間的溝通，且可以降低下一個發展周期的不確定性。在傳統的軟體發展周期可分為分析、設計、實作、測試等階段。

而就目前所現有的自動化軟體測試工具，大多是以程序語言為導向，無法完全

應用在物件導向軟體發展上。因此有必要發展一套適合於物件導向軟體的自動化測試工具。而我們又希望此自動化工具可以在物件導向軟體的設計階段就可以應用且其所產生的結果，可以做為錯誤的更正和下一個發展周期可以依循的標準。本計劃即根據此目的，發展出一套適用於物件導向軟體的發展中的設計階段，有效偵測類別層次的繼承性關係的重覆繼承性所反映出的複雜度。並據以建立其軟體測試的理論架構，其最終目的就是建立一套有效的自動化軟體測試工具，使軟體設計者能在物件導向軟體發展的設計階段，即可利用此工具偵測出不適當的結構，能及早作修改，藉以幫助軟體的設計，降低軟體的發展成本。

在本計劃中，基於物件導向程式設計觀念改變了軟體架構及程式的動態行為，同時也使得測試及衡量方式更為困難。從測試者的觀點來看，一個物件導向軟體系統可以分為四個程度的測試抽象化，分別為程序〔Routine〕、類別〔Class〕、聚集〔Cluster〕與系統〔System〕四個層次。在程序層次中，考慮程序中與處理資料的演算法之程序碼，是最低階的測試方式，即是所謂的單元測試〔Unit Testing〕；在類別層次中，考慮被包裝在類別中的程序與資料間的交互作用；在聚集層次中，考慮一組相關類別之間其互相交連的運作情形；系統層次，考慮所有程序碼，類別及主程式結合的整體測試方法。

在程序的測試階段，面對的是一個一個獨立的程序及資料，因此傳統的結構測試方法仍然可以應用。由於類別是物件導向程式的組成基礎，也是與傳統程式最大不同處，因此類別測試正是物件導向軟體系統的測試重點，類別程式在測試上造成下列幾個方面的影響：

1. 類別測試的問題。
2. 繼承和重覆使用問題。
3. 朋友類別的問題。
4. 動態繫結的問題。
5. 記憶體使用的問題。

我們在計畫中提出一自動化類別階層解析法，來解決存在於繼承和重覆使用上的問題。以期能夠測試類別之間異常的繼承錯誤。例如，隔代繼承、成員重覆的問題與成員名稱重覆的問題。

三、結果與討論

本計畫完成一套自動化軟體測試工具，根據程式設計師完成的C++程式及本計畫提出的URI測試準則，標示出該程式的URI圖及其複雜度，供軟體工程師們參考並及早發現與改進缺失。

參考執行範例如附圖。

四、計畫成果自評

本計畫對物件導向程式設計之軟體提出新而有用的測試及衡量方法，並將之予以實作成一工具軟體。因此，在學術研究上，我們有突破及成就。在對軟體工業界上，我們有提供軟體工程師們用以提高軟體開發的工具，可以增進軟體發展的品質與效能。對學生的訓練方面，我們提供了學生從軟體測試的角度培養他們更犀利的程式寫作技巧及對物件導向程式設計的認識，也對資訊專業的訓練提供一個良好的機會。

五、參考文獻

[1] Shyam, R. Chidamber and Chris, F. Kemerer, "Towards a Metrics Suite for Object-Oriented Design," OOPSLA'91 Proceedings,pp. 197-211, 1991.

[2] C. M. Chung, Timothy K. Shih, C. C. Wang, and M. C. Lee, "Object-Oriented Software Development Techniques – Combining Testing and Metrics," International Journal of Information and Management Sciences, Vol. 6, No. 4, pp. 17-34, December 1995.

[3] C. M. Chung, Chun-Chia Wang, and Timothy K. Shih, "A Hierarchy Testing of Object-Oriented Program Structure," Proceedings of the Seventh Software Engineering and Knowledge

Engineering, IEEE Computer Society, Maryland, U.S.A.,pp. 320-327, June 22-24, 1995.

[4] C. M. Chung, Chun-Chia Wang, Timothy K. Shih, and Jich-Yan Tsai, "A Metric of Inheritance Hierarchy for Object-Oriented Software Complexity," Fifth International Conference on Software Quality, Austin, Texas, U.S.A., October 23-26, 1995.

[5] C. M. Chung and M. C. Lee, "Inheritance-Based Metric for Complexity Analysis in Object-Oriented Design," Journal of Information Science and Engineering, No. 8,pp.431-447,1992.

[6] H. E. Dunsmore and J. D. Gannon, "Data reference: an Empirical Investigation," IEEE Computer, pp.50-59,December 1983.

[7] W. A. Horrison and K. I. Magel, "A Complexity Measure Based on Nesting Level," ACM SIGPLAN Notices, Vol.16, No.3,pp.63-74,March 1981.

[8] I. Jacobson, "Object-Oriented Software Engineering: A Use Case Driven Approach," Addison-Wesley, 1992.

[9] J. W. Laski and B. Koerl, "A data flow oriented programming testing strategy," IEEE Trans. on Software Eng., Vol.9,No.3,May 1983.

[10] M. Lorenz, "Object-Oriented Software Development: A Practical Guide," Prentice Hall, 1993.

[11] T. J. McCabe, "A Complexity Measurement," IEEE Trans. on Software Eng., SE2(4),pp.302-308,1976.

[12] T. J. McCabe, "Design Complexity Measurement and Testing," CACM, Vol.32,pp.1415-1425,Dec. 1989.

[13] B. Meyer, "Object-Oriented Software Construction," Prentice Hall, 1988.

[14] S. C. Ntafos, "On Required Element Testing," IEEE Trans. on Software Eng., Vol. SE-10,No.6,pp. 795-803,Nov. 1984.

[15] D. E. Perry and E. G Kaiser, "Adequate Testing and Object-Oriented Programming," JOOP,2(5),pp. 13-19, 1990.

[16] S. Rapps and E. J. Weyuker, "Selection software test data using data flow information," IEEE Trans. on Software Eng., Vol. SE-11, No. 4, pp. 367-375, April 1985.

[17] M. D. Smith and D. J. Robson, "A framework for testing object-oriented programs," JOOP,pp. 45-63, June 1992.

[18] C. Berge, "Graph and Phypergraphs," (Amsterdam: North-Holland), 1973.

[19] W. Li and S. Henry, "Object-Oriented Metrics that Predict Maintainability," J. Systems and Software ,pp.111-122, 1993.

[20] Eun Mi KIM, Ok Bae Chang, Shinji KUSUMOTO, and Tohru KIKUNO, "Analysis of Metrics for Object-Oriented Program Complexity," Proceedings of COMPSAC'94, IEEE Computer,pp.201-207, Nov. 9-11, 1994.