行政院國家科學委員會專題研究計畫成果報告

以工作元分解機制對並行軟體測試與衡量之研究
Preparation of NSC Project Reports

計畫編號：NSC 87-2213-E-032-009
執行期限：86 年 8 月 1 日至 87 年 7 月 31 日
主持人：王英宏　　淡江大學資訊工程系

一、中文摘要

軟體測試與衡量是兩個軟體可靠度與品質保証的方法。在循序軟體方面，軟體測試與衡量已有相當成熟的方法論與工具來驗證被測試程式的正確性。近幾年來並行軟體的興起相對地引導出新的測試問題與困難，非傳統循序軟體測試技術所能解決的。其中的困難之一是同一個並行程式在一組相同的資料時可以有許多不同的執行實例產生。已有許多並行程式測試的方法被提出來，目的在解決執行結果的可控制性及可決定性。但是極少有並行程式測試是從工作元之間的觀點來討論軟體測試。然而；並行程式都具有相同的特徵，就是一個並行程式是由許多明確的平行計算單元（即工作元，tasks）組成，而且這些工作元之間會透過一種「會合型式」的機制（Rendezvous-Style mechanism）來互相傳遞訊息。而事實上，扣除掉這種工作元間的通訊功能，每個工作元可以視為一個循序的程式，因此本研究的目的就是在透過工作元分解機制來討論並行軟體的測試問題，並以工作元間通訊的觀點，提出並行軟體測試準則（criterion）。而且透過這種工作元間的通訊機制（Mechanism），我們亦可同時提出對並行軟體複雜度的評量方法。最後本研究計畫將針對所提出的測試標準與衡量方法建構一個工具軟體（CASE tool），來協助程式設計師對已開發的並行程式進行測試與複雜度衡量。

關鍵詞：並行程式、軟體測試標準、軟體複雜度、Ada 程式語言、會合。

Abstract

Software testing and metrics are two important approaches to assure the reliability and quality of software. Testing and metrics of sequential programs have been a fairly sophisticated process, with various methodologies and tools available for use in building and demonstrating the correctness of a program being tested. The emergence of concurrent programming in the recent years, however, introduces new testing problems and difficulties that cannot be solved by testing techniques of traditional sequential programs. One of the difficult tasks is that concurrent programs can have many instances of execution for the same set of input data. Many concurrent program testing methodologies are proposed to solve controlled execution and determinism. There are few discussions of concurrent software testing from the inter-task viewpoints. Yet, the common characteristics of concurrent programming are explicit identification of the large grain parallel computation units (tasks), and the explicit inter-task communication via a rendezvous-style mechanism. In this paper, we focus the testing view on the concurrent programming through task decomposition. We propose four testing criteria to test a concurrent program. Programmer can choose an appropriate testing strategy depending on the properties of concurrent programs. Associated with the strategies, four equations are provided to measure the complexity of concurrent programs.

Keywords: Concurrent programs, software

testing criterion, software complexity, Ada language, rendezvous.

二、緣由與目的

軟體測試與軟體衡量是軟體發展生命週期中兩個非常重要的技術，軟體測試與衡量是軟體可靠度與軟體品質保證的兩個重要的方法，而軟體測試與衡量的相關技術在傳統的循序執行程式方面的技術與方法已經相當成熟，而且有許多不同的方法論工具可供選用。然而，在過去十年來，並行執行程式的測試被廣泛的討論，結果產生了一些傳統循序執行程式除錯技術無法解決的新問題，在此研究中，我們將探討測試並行執行程式會遭遇到的問題，並從工作元及工作元之間的角度提出新的測試策略。

並行執行程式是由可以並行執行的元件所組成，擁有並行執行能力的程式有許多好處，例如減少程式執行時間，而且對於作業系統、即時系統、資料庫系統等可以有較明顯的符號及觀念來撰寫。但是，並行執行程式的測試工作則是困難的，由於並行執行程式的不可決定性，同一個並行執行程式對同一組相同的輸入資料可以有許多不同的執行實例，雖然或可對一個不可決定的並行執行程式重複執行同一組輸入資料，但仍不足以審視所有實例執行的情形，最差的情況是只有一個錯誤存在於某一執行實例，而這個執行實例卻在反覆執行中未被測試到。因此，任何實際並行執行程式測試的方法必須能夠對應一組輸入資料，而查出一個以上的執行實例以找出可能的錯誤。

測試循序執行程式已被認為是一件相當複雜的程序，目前已有不同的方法論及工具來確立及展現待測試程式的可靠度。然而，近幾年來平行執行程式的興起，已經產生出許多一般循序執行程式測試技術所無法解決的新測試問題，雖然已有許多根據不同技術而發展出來的測試策略已被提出，但仍存在著一些問題，我們將在下一章節描述相關的研究工作。然而，卻少有研究論及從並行執行程式的工作元及工作元之間的通訊觀點來討論並行執行軟體的測試策略的。

並行執行程式語言的一個共同特徵是由明顯的平行計算單元（稱為工作元，tasks）所組成，然而這些工作元之間藉由一「會合型式」的機制作為工作元間的通訊。一些現有的並行執行程式設計語言如 HAL/S、CSP、Ada 及 PCF FORTRAN 等，都有支援這種能力。而扣除這種工作元間通信的能力，每一個工作元本身與一個循序執行程式的結構沒什麼兩樣。雖然任何一種使用會合式同步的程式語言都能產生可用的結果，為了提供往後研究特定的基礎，我們將以 Ada 語言做為我們的例子。Ada 允許一個程式指定任何平行執行的工作元個數，工作元之間的同步與通訊稱為「會合」，這個「會合」的觀念包含了行程間的同步與通訊，兩個行程的溝通由同步開始，然後在繼續它們各自的工作前交換資訊，這個交換資訊的同步或通訊動作就叫「會合」，因此我們將軟體測試的焦點放在平行執行程式的會合上，我們將依據會合機制提出一些測試策略及衡量法則。此研究計劃將把焦點放在 Ada 程式語言，並且假設並行執行單元間工作元的變數將不會共用。

這篇研究報告的架構如下：第二節介紹並行執行程式測試的相關研究，第三節我們將提出根據會合機制所發展出的四種測試準則，第四節將提出涵蓋準則架構（coverage criterion hierarchy）及相關證明。根據會合機制，我們更進一步提出四個可以計算並行執行程式複雜度的公式，這些公式將在第五節介紹，第六節介紹根據研究成果開發的軟體工具，第七節是這篇研究報告的結論，以及我們未來的研究工作。

三、結果與討論

近來，並行/平行測試是非常受重視的。測試並行/平行程式被視為更加的困難，因為並行程式通常是不可決定性的。因此，許多並行測試策略根據其並行程式的特性被提出來。

並行程式與循序程式的比較，主要特徵之一就是會合。我們提出了對並行程式測試的會合測試機制。在我們的研究中我們提出了以會合形式為基礎的並行/平行程式的四個測試準則。根據對工作元 entry calls 及 entry acceptances 特性的分析，程式設計師可選適當的測試策略去對並行程式除錯。假設執行並行測式之前，個別工作已經以循序程式設計技術完整的測試過。我們也提出對四測試準則的涵蓋準則階層架構並證明涵蓋階層架構的正確性。我們提供了以會合為基礎的四個等式以用來對並行/平行程式的測量。除此外，我們對以會合形式複雜度的並行程式設計做了兩項建議。

我們也依此研究成果轉化為具體的工具軟體，開發並行程式的軟體工程師們可以利用此工具軟體對其所發展的軟體進行分析，本工具軟體將提供可行的測試測略及複雜度的衡量結果。對於並行軟體的開發品質與可靠度有極大的幫助。

相關的執行範例見附圖所示。

四、計畫成果自評

本計畫對並行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並行程式設計的認識，也對資訊專業的訓練提供一個良好的機會。

五、參考文獻

[1] M. E. Conway, "Design of a Separable Transition Diagram Compiler," CACM, pp. 396-408

[2] Suresh K. Damodaran-Kamal and Joan M. Francioni, "Nondeterminacy: Testing and Debugging in Message Passing Parallel Programs," ACM SIGPLAN Notices, pp. 118-128, Dec., 1993

[3] Suresh K. Damodaran-Kamal and Joan M. Francioni, "Testing Races in Parallel Programs with an OtOt Strategy," Proceeding of the 1994 International Symposium on Software Testing and Analysis(ISSA), also ACM Software Engineering Notices, special issue, pp. 216-227, Aug., 1994

[4] DoD, "Preliminary Ada Reference Manual," SIGPLAN Notices, Vol. 14, No. 6, Part A, Jun., 1980

[5] Narain Gehani, "Ada: Concurrent Programming," 2nd Edition, AT&T Bell Lab., Silicon Press, 1991

[6] C. A. R. Hoare, "Communicating Sequential Processing," Communication of ACM, Vol. 21, No. 8, pp. 666-677, Aug. 1978

[7] T. J. LeBlanc and J. M. Mellor-Crummey, "Debugging Parallel Programs with Instant Replay," IEEE Trans. on Computers, C-36, No. 4, pp. 471-482, Apr., 1987

[8] Louise E. Moser, "Data Dependency Graphs for Ada Programs," IEEE Trans. on Software Engineering, Vol. 16, No. 5, pp. 498-509, May, 1990

[9] Parallel Computing Forum, "PCF Parallel FORTRAN extension," FORTRAM Forum, vol. 10, No. 3, special issue, Sept. 1991

[10] K. C. Tai, R. H. Carver, and E. E. Obaid, "Debugging Concurrent Ada Programs by

Deterministic Execution," IEEE Trans. on Software Eng., Vol. 17, No. 1, pp. 45-63, Jan. 1991

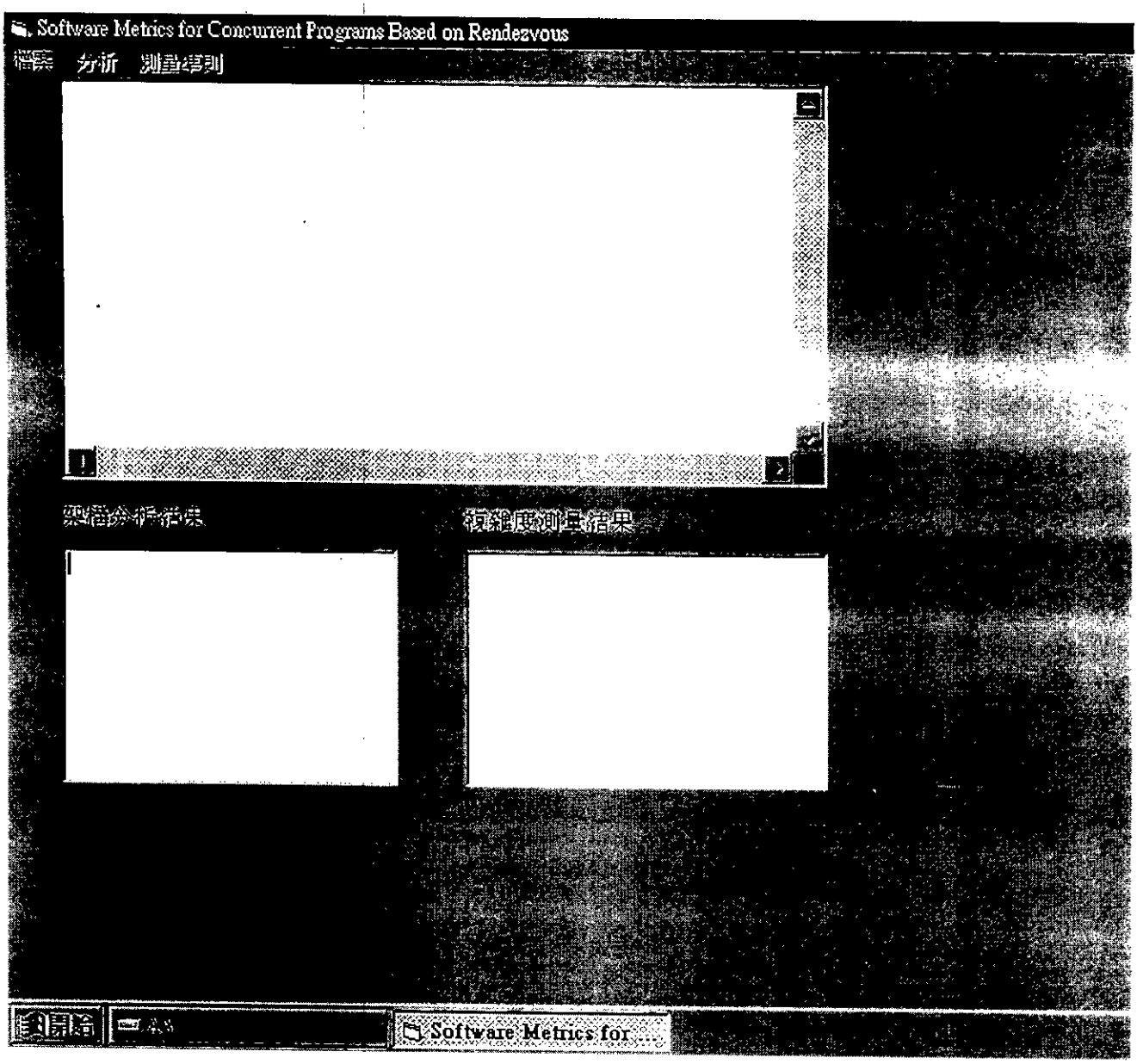
[11] R. N. Taylor, "A General Purpose Algorithm for Analyzing Concurrent Programs," CACM, pp. 362-376, May, 1983

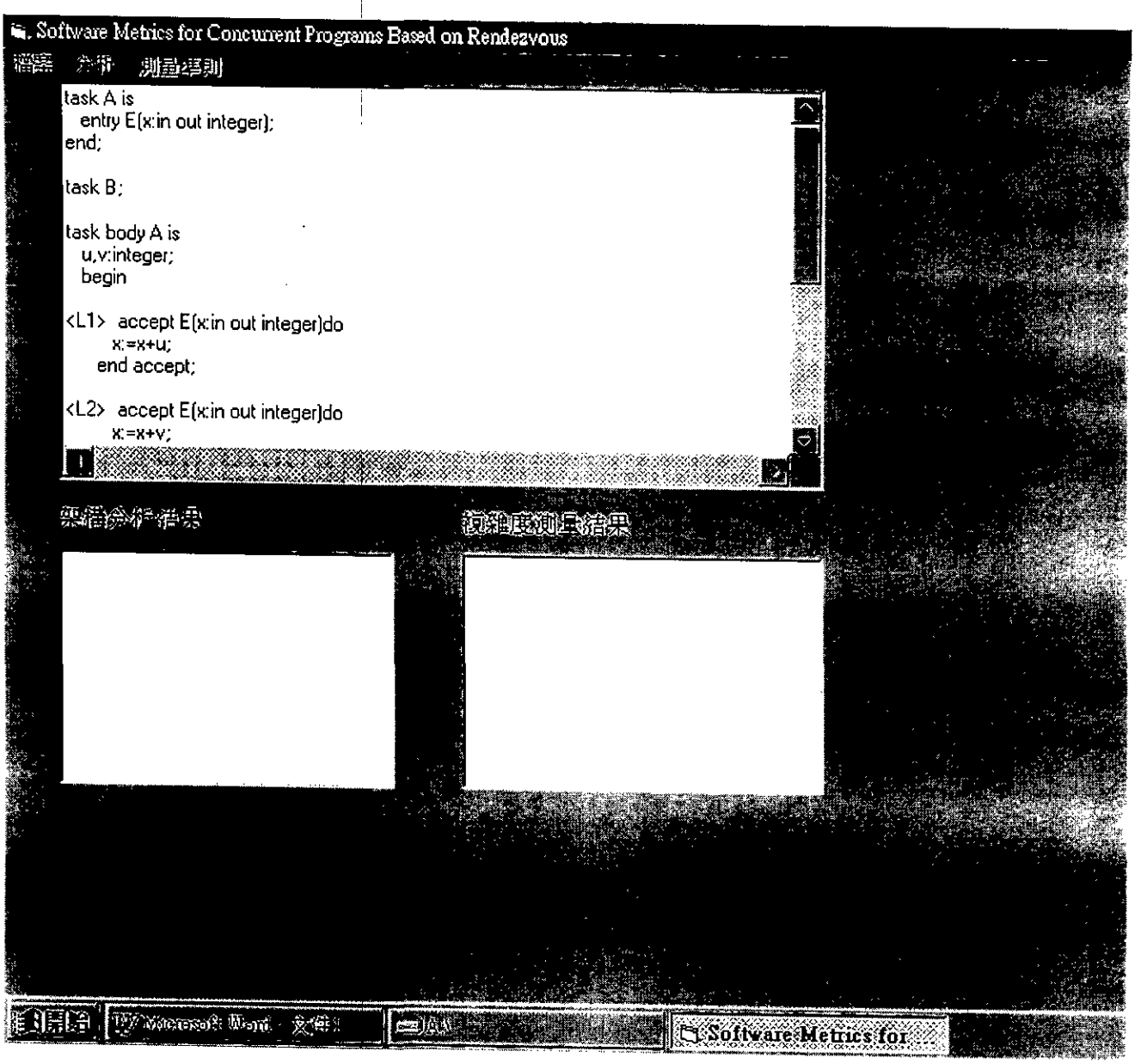
[12] R. N. Taylor, D. L. Levine and C. D. Kelly, "Structural testing of Concurrent Programs," IEEE Trans. on Software Eng., Vol. 8, No. 3, pp. 206-215, March, 1992

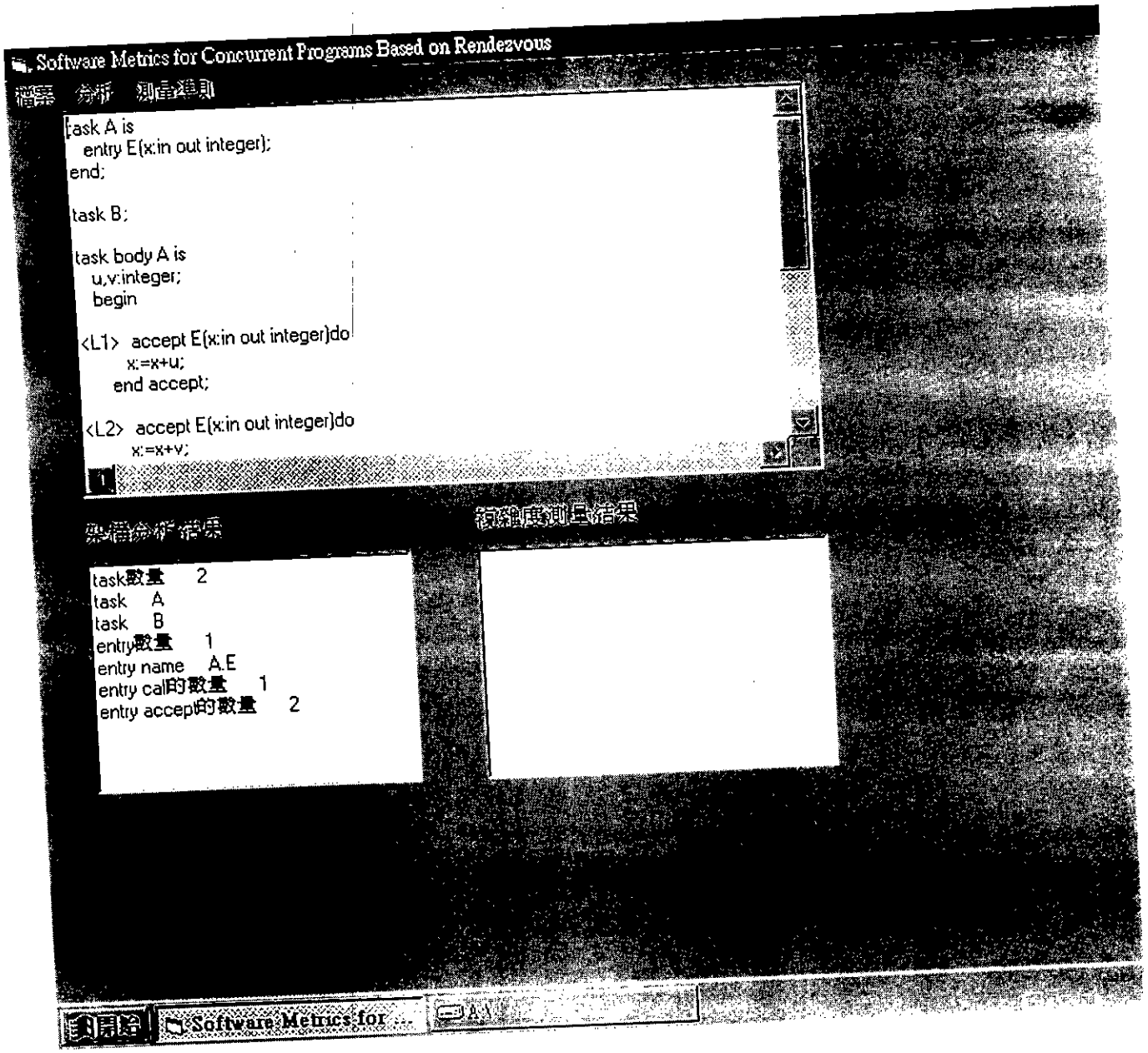
[13] J. J. Tsai, K. Y. Fang H. Y. Chen and Y. D. Bi, "A Noninterference Monitoring and Replay Mechanism for Real-time Software Testing and Debugging," IEEE Trans. on Software Eng., Vol. 16, No. 8, pp. 897-915, Aug., 1990

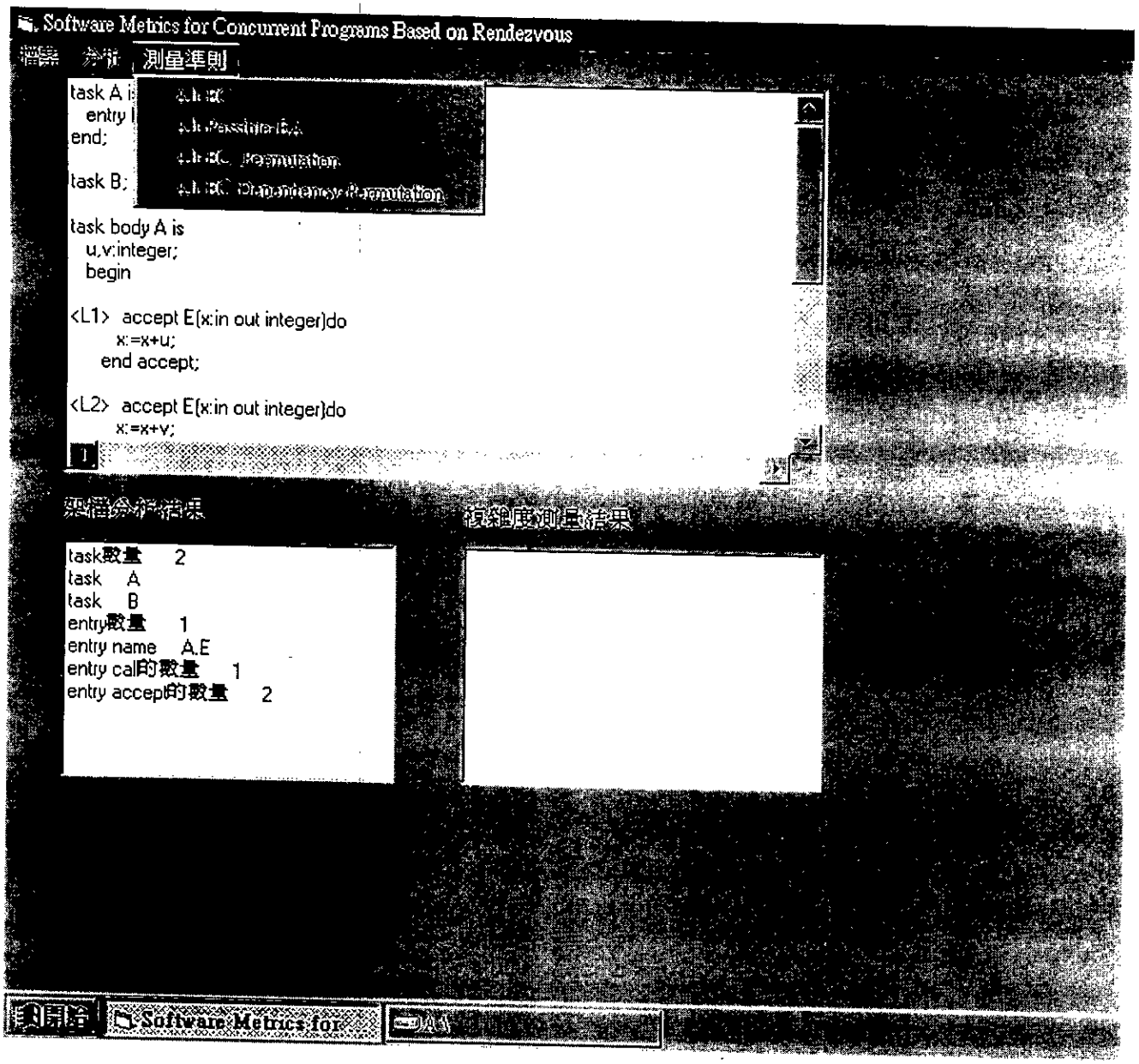
[14] S. Weiss "A Formal Framework for The Study of Concurrent Program Testing," In Proceedings of the 2nd Workshop on Software Testing, Analysis, and Verfication, pp. 106-113, July, 1988

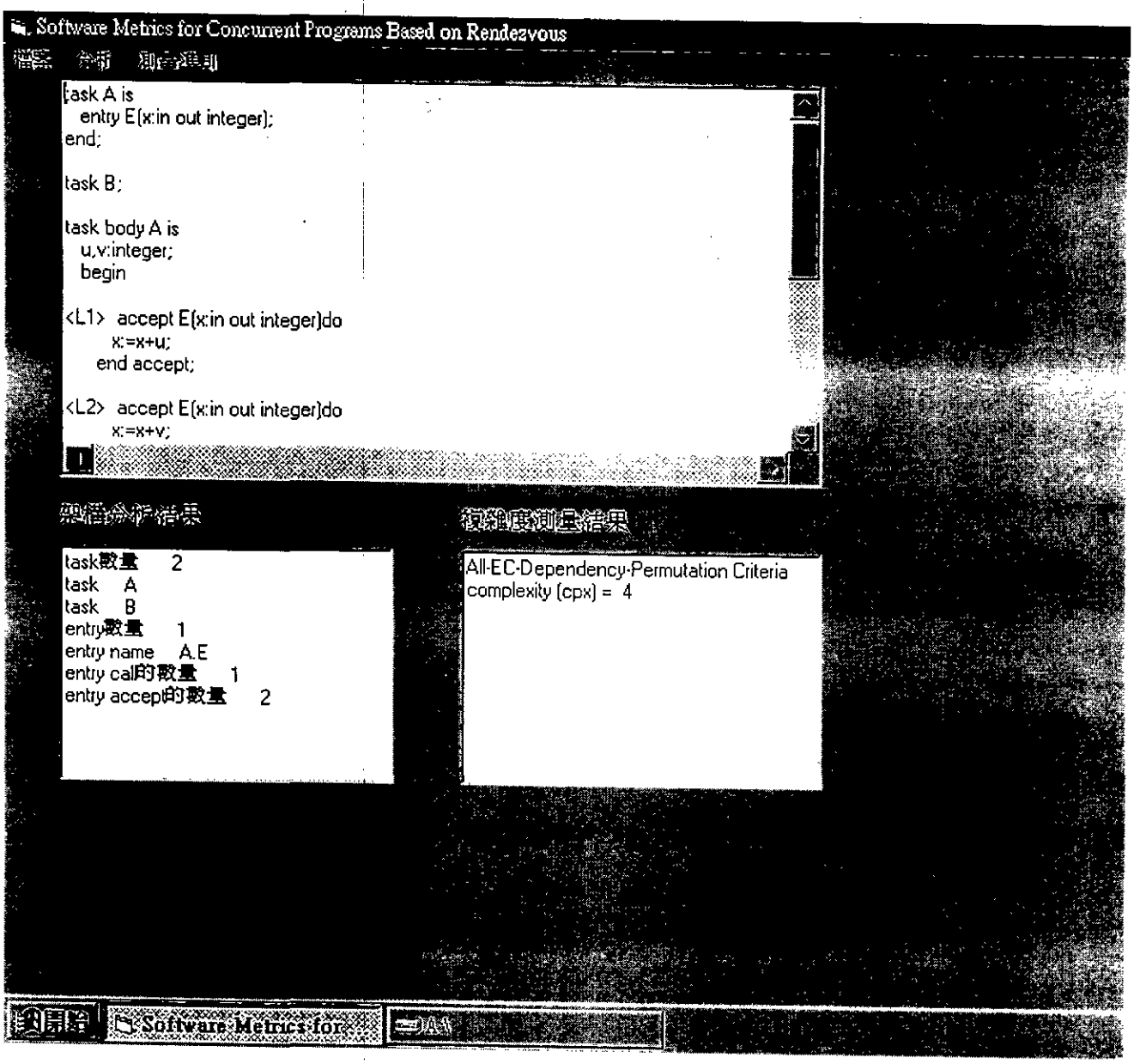
[15] S. Morasca and M. Peeze, "Using High Level Petri Nets for Testing Concurrent and Real Time Systems, " In Real-Time Systems: Theory and Application, pp. 119-131, 1990


Software Metrics for Concurrent Programs Based on Rendezvous
檔案 分析 測量準則
程式分析結果
複雜度測量結果
開始
Software Metrics for ...


Software Metrics for Concurrent Programs Based on Rendezvous
task A is
entry E(x:in out integer);
end;
task B;
task body A is
u,v:integer;
begin
<L1> accept E(x:in out integer)do
x:=x+u;
end accept;
<L2> accept E(x:in out integer)do
x:=x+v;
Software Metrics for


Software Metrics for Concurrent Programs Based on Rendezvous
檔案 分析 測量準則
task A is
entry E(x:in out integer);
end;
task B;
task body A is
u,v:integer;
begin
<L1> accept E(x:in out integer)do
x:=x+u;
end accept;
<L2> accept E(x:in out integer)do
x:=x+v;
end;
1
靜態分析結果
task數量 2
task A
task B
entry數量 1
entry name A.E
entry call的數量 1
entry accept的數量 2
Software Metrics for ...


Software Metrics for Concurrent Programs Based on Rendezvous
檔案 分析 測量準則
task A is
entry
end;
task B;
task body A is
u,v:integer;
begin
<L1> accept E(x:in out integer)do
x:=x+u;
end accept;
<L2> accept E(x:in out integer)do
x:=x+v;
end;
Permutation
Transitive Permutation
靜態分析結果
複雜度測量結果
task數量 2
task A
task B
entry數量 1
entry name A.E
entry call的數量 1
entry accept的數量 2
開始
Software Metrics for


Software Metrics for Concurrent Programs Based on Rendezvous
檔案 分析 測度準則
task A is
entry E(x:in out integer);
end;
task B;
task body A is
u,v:integer;
begin
<L1> accept E(x:in out integer)do
x:=x+u;
end accept;
<L2> accept E(x:in out integer)do
x:=x+v;
語法分析結果
task數量 2
task A
task B
entry數量 1
entry name A.E
entry call的數量 1
entry accept的數量 2
複雜度測量結果
All-EC-Dependency-Permutation Criteria
complexity (cpx) = 4
開始
Software Metrics for...