行政院國家科學委員會補助專題研究計畫成果報告

製造業管理資訊系統的複雜度評估-一種馬可夫分析模式

A Stack-based Markov Model for the Complexity Measurement of Production Management Information Systems

計畫類別： ■**個別型計畫** □**整合型計畫**

計畫編號：NSC89－2213－E－343－001－

執行期間： 88 **年** 8 **月** 1 **日 至** 89 **年** 7 **月** 31 **日**

計畫主持人： **駱至中**

共同主持人： **王鄭慈**

本成果報告包括以下應繳交之附件：

□**赴國外出差或研習心得報告一份**

□**赴大陸地區出差或研習心得報告一份**

□**出席國際學術會議心得報告及發表之論文各一份**

□**國際合作研究計畫國外研究報告書一份**

執行單位：南華大學資訊管理學系暨研究所

中 華 民 國 89 **年** 9 **月** 25 **日**

行政院國家科學委員會專題研究計畫成果報告

製造業管理資訊系統的複雜度評估-一種馬可夫分析模式

A Stack-based Markov Model for the Complexity Measurement of Production Management Information Systems

計畫編號：NSC 89-2213-E-343-001
執行期限：88 年 8 月 1 日至 89 年 7 月 31 日

主持人：駱至中、王鄭慈　　南華大學資訊管理學系暨研究所
E-Mail: locc@ mail.fgu.edu.tw、 ctwang@mail.fgu.edu.tw

一、中英文摘要

製造管理資訊系統開發及維護成本的多寡與系統複雜度有顯著的正向關係，資訊系統的開發者與使用者因此需要如軟體複雜度評估（software metric）這種可以用來預估管理資訊系統開發及維護所需的時間、人力與經費及其可靠性的工具。運用古典資訊理論的觀念，在堆疊式的馬可夫分析模式中，資訊系統的資訊內容（information content）或系統開發者的熵（entropy）已被證實可用來代表該系統的複雜程度。依據此一觀點及我們先前相關研究的結果為基礎，在本研究計畫中，我們建立了一個更合理而且完善的管理資訊系統之複雜度評估模式。 這個系統複雜度評估模式兼顧到因程式控制流程以及程式大小所產生的系統複雜度。此一模式可用為管理資訊系統或一般應用軟體開發前用來預估系統開發所需各項成本之有效工具。除了理論與實際的管理資訊系統複雜度評估及分析模式建立外，我們並以不同之管理資訊系統程式組作為測試樣本，驗證了這個新的複雜度評估模式。

關鍵詞：製造管理資訊系統、資訊系統評估、系統分析與設計、軟體複雜度評估、資訊理論、馬可夫模式

Abstract

There is no doubt that information systems with high efficiency in handing complex information can help manufacturing companies gain more advantage in global competition they are facing today. Software metric is an important and active topic in system engineering and management, in which numbers that are extracted from the software itself are used to indicate the conceptual complexity and to predict the maintenance time, cost, and reliability of the software. Since management information systems are also software, software metric is certainly a handy technique to measure the complexity of information systems for selected evaluation tasks. The Stack-Based Markov (SBM) model is a new-sprung approach for measuring complexity of information products. In the SBM model, the generation of a new symbol is accompanied by transformation of the top of the stack, and the probabilities for the next symbol depend on the top of the stack. So, the information contents of programs or entropies of development sources can be used as indicators of system complexities. In this research, based on the SBM model, we have developed a reasonable analyzing model and a new software metric to measure complexity of management information systems. This new approach takes the complexity of control structure of program and the complexity of program blocks into consideration. We also propose a new and proficient complexity measurement for blocks of programs, in which the complexities for control flow of expressions, are considered. Several sets of programs of real information system are also used to validate the proposed model.

Keywords: management information systems, information system evaluation, system analysis and design, software metric, information theory, Markov analysis model

二、計畫緣由與目的

In many managerial fields, numerical descriptions extracted from products or business and manufacturing processes are helpful for selecting better products or improving business and its processes. In software engineering, software metrics extractable from software or development processes can be recognized as their conceptual complexities. These conceptual complexities can also be used to predict the cost or reliability of products and processes. In general, software metrics can be classified into two categories: software product metrics and software process metrics [9]. Software product metrics are measures of software products, such as source codes and design documents. Software process metrics are measures of the software development process, in which they focus on the effort of development such as time and methodology used. The research described in this paper **proposes a new** software product metric **based on the information theory and the Stack-based Markov model. This new software metric can be used to reflect not only the size complexity but also the structure complexity of cited software. Experimental tests are performed to validate this proposed software metric and the result shows that the proposed metric is a reasonable software complexity measure and** is certainly a vigorous tool to measure the complexity of manufacturing information systems for certain evaluation tasks.

三、研究方法、結果與討論

There are many related software product measures proposed by previous researchers. Such measures may be simply count the lines of computer programs for complexity measurement, but the definition of line is not well defined [1]. Another well known textual measure for complexity measurement is the "software science" proposed by Halstead, based on both length in operators and operands and vocabulary in unique operators and operands [5]. All textual measures do not consider the structure complexities of the software that they are evaluating. The most famous structure measure is the *cyclomatic number* proposed by McCabe [8], in which the complexity of a program is determined by the structure of control flow graphs of that program (denoted by G). The cyclomatic number of the control flow graph G, $v(G)$, is computed as $v(G)=e-n+2$, where the notation e is used to designate the number of edges, and the notation n is used to designate the number of nodes in the graph G. The cyclomatic number is easy to calculate and is widely accepted, but it still has two major weaknesses: (1) ignorance of the nesting complexity and (2) exclusion of the inside complexities of blocks of codes. Based on McCabe's cyclomatic number, many measures are proposed later by Evangelist and by Harrison and Magel to represent the complexities for nesting levels of control flow [4, 6], but they still not include the complexities of program blocks.

The first work that applies information theory to deal with symbols that carry information is done by Shannon [10]. In the Markov model with traditional linear bounded memories, each symbol depends on its preceding symbols. However, in real programs, the generation of symbol may not depend on its preceding one in all cases. For example, considering the structure flow of a program, in a case where there are two nesting "if" structures in the program, the second "end-if" symbol is in fact paired with the first "if" symbol instead of the second "if" symbol. Hence, it is clear that an information source with traditional memory is not adequate to reflect or to parse the syntax of programming languages. Edwards proposed an information source with a pushdown stack rather than bounded memory [2, 3]. They defined a *stack-based Markov (SBM)* source, where the generation of a new symbol is accompanied by transformation of the top of s stack, and the probabilities for the next symbols generated depend on the top

of the stack. The information content of a symbol is then determined by its conditional probability relative to the top of the stack. The information content of a program P is calculated as the summation of logarithms of the inverse of generated symbols' conditional probabilities, and is described as Eq. (1). The entropy of a source, S, is the average information content of each symbol, and is calculated as shown in Eq. (2).

$$I(P) = \sum \log \frac{1}{(x_j \mid c_i)} (bits) \quad (1)$$

$$H(S) = \sum_i p(c_i) \sum_j p(x_j \mid c_i) \log \frac{1}{p(x_j \mid c_i)} (bits) \quad (2)$$

In previous two equations, x_j is used to represent the *j-th* symbol of the program and c_i is used to denote the stack top of the SBM. Many SBM models have been proposed and validated to include expressions, data dependencies, and object oriented design [3, 7, 11, 12, 14, 15].

3.1 The Proposed Model

In Wang's previous research [12, 13], SBM models are proposed for measuring the expressions and programs complexities. Although they are very complicate and quite difficult to understand, there is an important concept are proposed, the structure complexity of expression. As we know, a program consists of two part, control structure and expression parts. And, the main part of a program is the expression part of the computer program. All the software metrics for measuring expressions complexities focus on the size complexity only. The SBM model for expressions includes both structure and size complexities for expressions. The structure complexity for an expression is determined by the precedences and associativities of operators in the expression. For example, the two expressions, "a+b*c" and "c*b+a," have the same number of operators and operands, as well as the size complexity. But they have different order of operators and operands, that is, they have different evaluating order. And the former one is more complicate than the latter one, when evaluating and understanding them.

In order to understand the concept of structure complexity for an expression easily, the above expressions in example can be rewritten as "(())" and "()()," by considering the evaluating order. Hence, all expressions are transformed to strings of balanced parentheses, and the SBM model for expressions could be transformed to the SBM model for balanced parentheses. In this SBM model, all operators and operands in expressions are transformed parentheses representatively by their precedences and associativities. Then using SBM model for balanced parentheses, we can easily get different complexities for these expressions and the complexity for the former one is higher than the complexity for the latter one.

In this research, we propose a new SBM model for programs that includes structure and block complexities. The block complexity consists of size as well as structure complexities for expressions. And this new SBM model is much simpler than Wang's previous SBM models. First of all, we assume the control structure symbols and the expression symbols have the same weight in programs. Then, we convert all the expressions in a computer program into streams of parentheses, and combine them with the SBM model for balanced parentheses. Therefore programs are simplified to streams of parentheses, and the basic unit of programs is the parenthesis. In this proposed SBM model, the problem of block complexity does not exist. This new SBM model is shown in Table 1.

Using the proposed SBM model for balanced parentheses to analyze practical computer programs, we found a new problem that this SBM model could not be used to distinguish the nesting complexity between strings of parentheses. Taking expressions

"a=a*b+c+d" and "a=c+a*b+d" as example, they are equivalent to the strings of parentheses "(()()())" and "((())())". According to the calculation of the new SBM model, both of them get the same conditional probabilities p and q, and have the same entropy and information content. It is clear to see that they should have different entropies, and the second expression is more complicate than the first one. In this case, the new SBM model is not yet perfect. In order to solve this problem, the SBM model is modified as described in Table 2. As shown in Table 2, we add a counter to count the number of open parentheses pushed in the stack, and the counter is initialized to 0. When pushing stack action occurred, the counter is increased by 1; and when popping stack action occurred, the counter is decreased by 1. After these adjustments, the nesting problem is solved.

Table 1. The Proposed SBM Model for Balanced Parentheses

Stack Top	Next Symbol/Stack Action		
	(/push (	) /pop (	$ /none
Empty	p	0	1-p
(	q	1-q	0

Table 2. Modified SBM Model for Balanced Parentheses

Stack Top	Next Symbol/Stack Action		
	(/push (++counter	) /pop (--counter	$ /none
Empty	p	0	1-p
$(_1$	q_1	$1-q_1$	0
$(_2$	q_2	$1-q_2$	0
…	…	…	…
$(_n$	0	1	0

Using this modified SBM model, we found that the entropy for the first expression is 0.58279(bits), and is 0.83442(bits) for the second expression. This result indicates that the second expression is more complicate than the first one. Based on this implementation, it is obvious that differences or complexities among expressions of programs can be easily distinguished by using our proposed model.

3.2 Experimental Results

In order to validate our SBM model, we construct three C programs as testing examples. Program 1 and 2 have the same size but different structures. Program 2 and 3 have the same structure but different sizes. These C programs are measured by this modified SBM model, and the results are shown in Table 3. The entropy for program 2 is higher than the entropy for program 1, due to the difference between structure nesting complexities. The entropy for program 3 is higher than the entropy for program 2, caused by the block complexity. The result shows that this SBM model can reflect the structure and block complexities correctly.

Table 3. Entropies for 3 Testing Examples

Name	Entropy
Program1	0.86278
Program2	0.88618
Program3	0.89173

Another validation test is also performed on a genuine manufacturing information system for the management unit of an industry park in Taiwan. This system has 6 programs written in Visual Basic. The lines of code, number of symbols and the measuring result of the proposed SBM model analyzer for each program are summarized in Table 4. The result indicates that program Bprogram5 has the least size complexity but the highest program complexity. It means that program Bprogram5 has the highest structure complexity. This result also concludes that the program complexity depends not only on the structure complexity

but also the block complexity. This confirms that the proposed SBM model can include these complexities for programs exactly.

Table 4. Experimental Results of MISs

Name	Number of symbols	Total lines of code	Entropy
Bprogram1	18298	730	0.92322
Bprogram2	42605	1536	0.93062
Bprogram3	25880	987	0.94255
Bprogram4	32183	1082	0.93131
Bprogram5	11821	410	0.96767
Bprogram6	16742	517	0.85216

四、計畫成果自評

有關本計畫執行成果的評量, 可依下列角度分別列述：

- **就研究內容與原計畫相符程度而言，它們在內容和精神上是完全相同的。**
- **就達成預期目標情況來看，達成率也可稱是** 100%**達成。**
- **我們相信本計畫所提的方法更能充份反應管理資訊系統之複雜度，成為未來更具體地評估資訊系統與應用軟體的一個重要依據，同時作為降低系統開發及維護成本的參考指標。此一研究成果在學術上或應用上均有其價值，這些具體貢獻對於管理者在開發、擴充或選用任何管理資訊系統前的決策分析工作將大有幫助。**
- **至於本研究是否適合在學術期刊發表：本研究之部份研究成果將在年底發表於** IJIE2000 **國際學術研討會暨工工年會。 完整的研究結果也正進行實證資料補強和結構修改的工作，準備向國內外工業工程或糸統工程等相關學術期刊投稿。**

五、參考文獻

[1] Conte, S. D., Dunsmore, H. E. and Shen, V. Y. (1986), *A Software Engineering Metrics and Models*, Benjamin/ Cummings.

[2] Edwards, William R. (1990), "*Information Source Models for Software Analysis*," in *the 13th Minnowbrook Workshop on Software Engineering*, pp.8-17.

[3] Edwards, William R., Yang, Mingguey and Kim, Jong-Soo (1991), "*Application of the Stack-Based Markov Source to Software Analysis*," in *the 14th Minnowbrook Workshop on Software Engineering*, pp.44-62.

[4] Evangelist, Michael (1984), "*An Analysis of Control Flow Complexity*," in *Proceedings of IEEE COMPSAC-84*, pp. 388-396.

[5] Halstead, M. H. (1979), "*Advances in Software Science*," in *Advances in Computers*, Academic Press, vol. 18 pp.122-129.

[6] Harrison, Warren A. and Magel, Henneth I. (1981), "A Complexity Measure Based on Nesting Level," ACM SIGPLAN Notices, vol. 16, no.3, March, pp. 69-74.

[7] Holland, James A. (1993), "*A Stack based Object Oriented Development Model*," *Ph.D. Dissertation*, The University of Southwestern Louisiana.

[8] McCabe, Thomas J. (1976), "*A Complexity Measure*," in *IEEE Transactions on Software Engineering*, vol. SE-2, no.4, December, pp.308-320.

[9] Mills, Everald E. (1988), "*Software Metrics*," in *SEI Curriculum Module SEI-CM-12-1.1*.

[10] Shannon, C. E. (1948), "*A Mathematical Theory of Communication*," in *Bell Systems Technical Journal*, vol. 27, pp.379-423.

[11] Wang, Cheng-Tzu and Edwards, William R. (1991), "*An Implementation of the Stack-Based Markov Model on Pascal Code*," *Technical Report 91-4-1*, Center for Advanced Computer Studies, University of Southwestern Louisiana.

[12] Wang, Cheng-Tzu (1994), "*Stack-Based Markov Model Analysis of Expressions and Data Dependencies on Programs*," *Ph.D. Dissertation*, The University of Southwestern Louisiana.

[13] Wang, Cheng-Tzu (1997), "*A Probabilistic Model of Software Complexity Measurement*," in *Proceedings of the 8th International Conference on Information Management*, pp. 776-782.

[14] Yang, Mingguey and Edwards, William R. (1991), "*Experimental Study of the Stack-Based Markov Model*," *Technical Report 90-4-7*, Center for Advanced Computer Studies, University of Southwestern Louisiana.

[15] Yang, Mingguey and Edwards, William R. (1992), "*Stack-Based Markov Model for Imperative and Functional Languages*," *Technical Report 92-5-7*, Center for Advanced Computer Studies, University of Southwestern Louisiana.