

Design complexity metrics for the MASCOT 3 design method

J. Moses

*School of Computing and Information Systems, University of
Sunderland, Sunderland, SR1 3SD, UK*

Email: cs0jmo@uk.ac.sundp.orac

ABSTRACT

The importance of system design products has recently been recognised, [1], p.46, their maintainability, reusability and testability having been identified as being at least as important as that of the corresponding implemented code. Hence, investigation of complexity of software development products at architectural design (A.D.) and detailed design (D.D.) may be viewed as important because of the relationship between complexity and maintainability, see [2], [3], [4] and [1].

This paper demonstrates that, in general, there exists a highly significant positive correlation between the complexity of MASCOT 3 [5] modules (as measured by a suitable complexity metric) at architectural design (A.D.) and the complexity of the corresponding MASCOT 3 modules at detailed design (D.D.) (also measured by a suitable complexity metric). It follows that, to a large extent, the complexity of a MASCOT 3 module, at D.D. and implementation, can be determined from A.D..

Hence, armed with such knowledge it becomes possible to control complexity of MASCOT 3 modules and hence to assist in ensuring the maintainability of the implemented software, in addition to other quality factors dependent on module complexity.

INTRODUCTION

Currently information flow complexity metrics (IFC metrics), i.e. those proposed in [3] and [6], appear to be one of the most advanced and successful group of metrics to have been investigated, see [7] and [8], for the identification of software modules which may prove difficult to maintain. Further, a potential for the automatic measurement of information flow complexity metrics exists, [3].

Experiments conducted to validate the information flow complexity metric, due to Henry and Kafura [3] and later modified by Shepperd [6], have attempted to show the correlation between the metric and the number of changes to procedures [3] and system development effort [6]. The most satisfactory model for the metric currently appears to be that due to Shepperd in [6], see [1]. Several authors, including Pengelly et al. [8], Rombach [9], Henry and Kafura [3], state that the metric can be used from the architectural design stage to predict problems associated with module complexity. However, validation of the metric's correlation from measures taken at design with products of the software development process, e.g. errors in the implemented software, have not been undertaken and widely published. Consider [3] in which measures at the code stage were used to evaluate the metric for validation against the number of program errors in procedures, and Shepperd, in [6], who reverse engineered structure charts from code in order to achieve measures for the metric to validate against development effort. Although it is claimed that the IFC metric evaluated at A.D. can predict those software maintenance and implementation difficulties which are considered to be related to complexity, experiments to compare structural complexity at A.D. with complexity at D.D. have, also, not been widely undertaken and published (if at all). Hence, we cannot be certain, until such experiments have been completed and analysed, whether the metric at A.D. can predict, with accuracy, complexity at D.D., or even whether it can safely be used to predict development effort and other factors related to software complexity.

MASCOT 3 is one of several methods which provide a modular representation for software design, each module possessing clear and restrictive interfaces [10]. The technique used to represent architectural design in Shepperd's experiment [6] was the structure chart [2], [11], which does not possess such interfaces. MASCOT 3 enables some advantages over software development using the structure chart approach - global data structures are not allowed and each module possesses its own data structures which can only be accessed via the MASCOT3 access interface and its procedures. Thus, MASCOT 3 exhibits information hiding in addition to several other object-oriented features, see [12]. These MASCOT 3 features enable a simplification of the measurements needed to evaluate an information flow complexity metric. For example, absence of global data structures means counting global information flows, as required in [6], is unnecessary.

Further, because of MASCOT 3's high level of decoupling at A.D. [10] and the cohesive nature of its modules, modular complexity of a software design using MASCOT 3 should not increase dramatically from architectural design through to implementation. Thus, if this is so, then complexity as measured by an IFC metric at A.D. should be capable of accurately predicting complexity after the addition of program detail at D.D.. It is proposed therefore, that there should be a strong correlation between structural complexity measured at A.D. and complexity measured at D.D.. (The mechanism for measuring complexity at D.D. taking into account the increase in detail at D.D..)

THE INFORMATION FLOW METRICS FOR ARCHITECTURAL AND DETAILED DESIGN

The Mathematical Basis for the Models

The complexity of a procedure, according to Henry and Kafura in [3], 'is the complexity of the procedure code and the complexity of the procedure's connections to its environment'. This is a hybrid measure, see [1], containing two terms which require different measurement systems for three distinct attributes, i.e. number of data and control flows and number of lines of code. The hybridisation of metrics has been criticised by Fenton [1], and experimental evidence [3], [6] indicates that the procedure code factor should be excluded. Therefore, the internal complexity of a module, as estimated by code length, is not included in this metric and the term Structural Complexity is used in [6,7] (cf. Complexity of Interface [2]) to identify what the metric actually estimates. Further, Shepperd [6] does not count control flows.

The underlying model of the information flow complexity metric as a predictor of maintenance difficulties is:

$$\text{Complexity of Interface} = (\text{Fan_in} * \text{Fan_out})^2$$

where, in essence, fan_in is the number of data flows into a module and fan_out is the number of data flows out of the module.

Fenton in [1] indicates that a predictive measure must provide a mathematical model plus a set of prediction procedures for determining unknown parameters and interpreting results. The set of prediction procedures, etc. for Shepperd's model is given in [6]. The justification, given by Henry and Kafura [3], for the form of the model is that fan_in multiplied by fan_out represent "the total number of possible combinations of an input source to an output destination"; the square factor is used in the belief that "complexity is more than linear in terms of the connections which a procedure has to its

184 Software Quality Management

environment". The latter is justified by Henry and Kafura because Brook's law of programmer interaction [13] and Belady's formula for system partitioning [14] use a square factor.

Two major problems appear to detract from the models ability to model structural complexity: zero complexity and the square term. Ince and Shepperd [15] have discussed the validity of zero complexity for modules with either zero fan_in or zero fan_out, and did not find this problematic, since such modules tended to be relatively isolated from the rest of the system and therefore had little affect on the other modules. However, their assessment of the square factor was less favourable, and for them was not easily justified. Further, Brooke's law of programmer interaction [13] is not a pure square law: it is the number of combinations of n objects (programmers) taken two at a time, i.e. $n!/(2!(n-2)!) = n(n-1)/2$; and Belady [14] does not utilise a square term at all, unless, for example, the number of nodes and pairwise edges in a cluster are equal. Hence, the mathematical foundation for the squared term is dubious.

Metric for use at Architectural Design

The model for the metric at A.D. is based on Shepperd's metric [6] with some modifications for use with MASCOT 3. The most radical modification being the removal of the square term because of its dubious foundation. Further, the prediction procedures for the metric can be simplified because of MASCOT 3's modular structure and design constructs. Hence, the model at A.D. becomes:

$$\text{Complexity of Interface} = \text{IFC}_{AD} = \text{fan_in} * \text{fan_out}.$$

The prediction procedures are:

- fan_in = the number of unique flows into a MASCOT 3 IDA or activity.
- fan_out = the number of unique information flows out of a MASCOT 3 IDA or activity.
- a unique information flow = a local flow of data between two MASCOT 3 components excluding any duplicate flows.
- a local flow = an activity invokes an access procedure in an IDA and data passes from the IDA to the activity or an activity invokes an access procedure and data is passed to the IDA (data may also flow in both directions - due to the same invocation - in which case there are deemed to be two flows one in each direction).

- data = any data structures or structure, simple or complex, which pass(es) in the same direction following an invocation by an activity, i.e. separate data structures passed, in the same direction, as two or more parameters of an access procedure between modules are counted as one flow of information although separate parameters passing in opposite directions are enumerated separately - as an inflow and an out flow, see local flow.

Metric for use at Detailed Design

The basis for the model, as utilised by [6] and [3] is that $\text{fan_in} \times \text{fan_out}$ represents the total possible number of combinations of an input source to an output destination across the module interface and estimates the complexity of interface of a module. The measure at detailed design also uses this basis. This model is a derivative of the formula used in [3] to calculate the number of global flows in a module, the maximum number of paths of information possible through a module's data structure, i.e. all possible connections between the module's procedures and the data structure contained in the module. The metric used at D.D., like that at A.D., does not use the square term.

The formulae used to calculate *internal information flow complexity* (IFC_{DD}) for MASCOT 3 modules at detailed design are:

complexity of an IDA =

$$\sum_{ap} (\sum_k f_{ik} * WS_k + \sum_{DS} WS_{DS} * RS_{DS} + \sum_j f_{oj} * RS_j + \sum_p Pr_p f_{fo}) - (1)$$

complexity of activity =

$$\sum_k f_{ik} * WS_k + \sum_{DS} WS_{DS} * RS_{DS} + \sum_j f_{oj} * RS_j + \sum_p Pr_p f_{fo} - (2)$$

Where, for an activity f_{ik} is the number of flows into the k^{th} port; WS_k is the number of writes to a data structure from the k^{th} port; f_{oj} is the number of flows out of the j^{th} port; RS_j is the number of reads from a data structure to the j^{th} port; WS_{DS} is the number of writes to a data structure; and RS_{DS} is the number of reads from the same data structure; $Pr_p f_{fo}$ is the product of the flows into the p^{th} module process and the flows out of the process.

And where, for each access procedure, in an IDA, f_{ik} is the number of flows into the k^{th} window; WS_k is the number of writes to a data structure from the k^{th} window; f_{oj} is the number of flows out of the j^{th} window; RS_j is the number of reads from the j^{th} data structure; WS_{DS} is the number of writes to a data structure and RS_{DS} is the number of reads from the same data structure.

It should be noted that for both activities and access procedures a data structure may be a record, list, other complex data structure or simply a variable, i.e. no distinction is made between the complexities of the data structures themselves.

TRANSFORMATION

Detailed design of a MASCOT 3 module can be represented by a form of JSP diagram, as exemplified in [17], such diagrams describe the MASCOT 3 component access interfaces and procedures at detailed design. Further, within these diagrams data structures used in a component can be identified. A simple transformation from these JSP diagrams to a module data flow diagram can then be made, and evaluation of the formulae (1) and (2) is simplified by this data flow diagramming technique used to express data flow within a MASCOT 3 module.

DETAILED DESIGN STRUCTURE CHART

The notation for the detailed design structure chart is taken from the MOON method [17] which uses an extension of the JSP [18] notation for program structure. The main advantage of the notation is that consistency between MASCOT 3 modules and the structure chart can be maintained by annotating ports at relevant points in the JSP decomposition.

The purpose of the charts are to provide a means of representing the detail at D.D. and to assist in identifying data structures and information flows (identified through the port connections and the access interface names on the chart) which write to and read from them. Having identified the data structures and information flows module data flow diagrams can be constructed.

MODULAR DATA FLOW DIAGRAMS

For each activity module in a MASCOT 3 design a data flow diagram can be constructed using the module design from A.D. and the detailed design structure chart; for each IDA one or more module data flow diagrams may need to be constructed - one for each access procedure. The aim of these diagrams is to facilitate the calculation of the D.D. complexity metric (since the structure chart is unable to show internal data flow in a module).

Such diagrams show: data structures as data stores; data flow to and from ports on the interface of the module; reads and writes (flows across) all

identified data structures of a module; and also identify as diagram processes the internal actions or processes which read from/write to data structures - these processes are only small sections of what will eventually be implemented code. (External sources/destinations of data may be thought of as connected MASCOT modules.). Modular data flow diagrams show consistency with the MASCOT A.D. module and the D.D. structure chart through the port connections.

EXPERIMENTAL RESULTS

An experiment has been conducted, [19], to determine the relationship between the two metrics and the amongst the information derived from a statistical analysis of the data was the following:

- the Spearman rank correlation for the sum of the information flow complexity metrics at A.D. and D.D. was 0.857 at the 0.0005 level with $n=13$, see [20], for the complete data set, i.e. with no outliers removed.
- using linear regression with IFC_{AD} as the predictor and IFC_{DD} as the dependent variable the regression was $IFC_{DD} = 1.05 * IFC_{AD} + 7.27$. The residuals for the dependent variable showed no signs of non-normality and hence because 83.7% of the complexity variation was shown to be explained by the R-square adjusted value it can be safely assumed that the relationship is statistically significant, see [21].

CONCLUSIONS

This paper has outlined a technique and two metrics for the measurement of complexity during the design of MASCOT 3 modules. The metrics should be capable of use at the architectural and detailed design stages and assist in preventing module maintenance problems.

Further, the results of the experimental evaluation of these metrics will assist in demonstrating the benefits of the MASCOT 3 modular design method in controlling complexity from A.D. through the software life-cycle and hence in demonstrating MASCOT 3's ability to assist in producing maintainable design products and software.



REFERENCES

1. Fenton, N.E., *Software Metrics: A rigorous approach*, Chapman & Hall, 1993.
2. Stevens, W.P., Myers, G.J., Constantine, L.L., *IBM Systems Journal*, No. 2, 1974.
3. Henry, S., Kafura, D., Software Structure Metrics Based on Information Flow, *IEEE Trans. on SE* Vol SE-7 No. 5, Sept 1981.
4. Henry, J.A., Wake, S., Li, W., The Use of Metrics Throughout the Software Life-cycle, Computing Science and Statistics, Statistics of many parameters: curves, images, spatial models, *proc. 22nd Symposium on the Interface*, pp. 259-265, 1992.
5. Joint IECCA and MUF Committee, 1987, *The Official Handbook of MASCOT*, Version 3.1, Crown Copyright, UK.
6. Shepperd, M.J., Design Metrics: an empirical analysis, *Soft. Eng. Journal*, 5(1), 1990 310.
7. Shepperd, M., Ince, D., Metrics, outlier analysis and the software design process, *Journal of Information and Software Technology*, vol. 31 no.2, March 1989.
8. Pengelly, A., Benwood, H., Mihajlovich, R., Simons, C., Possibilities for Measurement Early in the Life-Cycle, *Proc. SQW 26/27 June 1990*, pp. 73-83.
9. Rombach, H.D., "A Controlled Experiment on the Impact of Software Structure on Maintainability", *IEEE Trans. Software Eng.*, March 1987, pp. 344-354.
10. Jackson, K., Design Methods for Real-Time Software, *IEE Colloquium on MASCOT and Related Issues*, London, 7th Dec. 1990.
11. Yourdon, E., *Modern Structured Analysis*, Prentice-Hall, 1989.
12. Moses, J., Object-oriented design and software construction methods, MASCOT3 and other Object-based methods: A comparison of their potential for producing quality in software, *3rd Software Quality Workshop*, Edinburgh, 5-6 July 1993.
13. Brooks, F.P., *The Mythical Man-Month: Essays on Software Engineering*, pp. 73-83, Addison-Wesley, 1975

14. Belady, L.A., Evangelisti, C.J., System s partitioning and its measure, *IBM Res. Rep. RC7560*, 1979.
15. Ince,D., Shepperd, M., An Empirical and Theoretical Analysis of an Information Flow based Design Metric, *Proc. European Soft. Conf.*, pp 11-16, 1989.
16. Shepperd, M., Measurement of structure and size of software designs, *Journal of Information and Software Technology*, vol. 34 No. 11, Nov. 1992.
17. Hull, M.E.C., O'Donoghue, P.G., Hagan, B.J., Development Methods for Real-Time Systems, *The Computer Journal*, Vol 34, No. 2, 1991.
18. Jackson, M.A., *System Development*, Prentice Hall Int., 1983.
19. Moses, J., *Phd. Thesis*, School of Computing and Information Systems, University of Sunderland, 1994.
20. Tate, M.W., Clelland, R.C., *Nonparametric and Shortcut Statistics in the Social, Biological and Medical Sciences*, Interstate Printers & Publishers Inc., Danville, Illinios, 2nd Print, 1959.
21. Schaefer, R.L., Farber, E., *Student Edition of Minitab 8 Users Manual*, Addison-Wesley.Benjamin/Cummings, Reading Massachusetts, 1992.